INGENIERIA TÉCNICA DE TELECOMUNICACIÓN: TELEMÁTICA

PROYECTO FIN DE CARRERA

Diseño e implementación de un sistema de personalización de guías de TV digital con detector de presencia

Autor: Álvaro Moscoso Cabrera

Tutora: Florina Almenares Mendoza

Julio de 2012

Título: Diseño e implementación de un sistema de personalización de guías de TV digital con detector de presencia

Autor: Álvaro Moscoso Cabrera

Directora: Patricia Arias Cabarcos

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

SECRETARIO VOCAL PRESIDENTE

Agradecimientos

Quiero agradecer por medio de estas líneas a todas aquellas personas que han dedicado parte de su tiempo a hacer posible la realización de este proyecto, dado que sin ellas, no habría sido posible.

A mis padres, por estar encima en todo momento, empujándome cuando más lo necesitaba. A mis compañeros, en los buenos y malos momentos vividos en estos maravillosos años de universidad, y en especial a Florina, por la paciencia y la dedicación con la que me ha premiado, y el aporte de nuevas ideas cuando el camino parecía cerrado. A todos ellos, muchas gracias.

Resumen

La TDT, o televisión digital terrestre, consiste en la aplicación de las tecnologías del medio digital a la transmisión de contenidos a través de una antena convencional. Aplicando dichas tecnologías se consiguen mayores posibilidades, como proveer un mayor número de canales o una mayor calidad de imagen y de sonido. De esta manera, se generan una serie de nuevos servicios, algunos de ellos a raíz de la evolución de los antiguos que ya existían en la televisión analógica, como es el caso del tradicional servicio de programación que nos ofrecía el teletexto y su actual sucesor, la EPG o guía electrónica de programas. Se trata de una organización sencilla de los canales que nos ofrece un distribuidor de televisión, permitiéndonos una exploración de todos ellos, suponiendo una gran ayuda para los usuarios. Pero las tradicionales EPGs aún siguen sin ofrecer una búsqueda efectiva, debido a que toman demasiado tiempo en que el espectador busque entre cientos de opciones su programa favorito.

El proyecto realizado consiste, por lo tanto, en diseñar e implementar un sistema de personalización de la actual televisión terrestre, de modo que cada usuario tenga asociado un perfil donde se almacenarán su identificador, sus gustos y preferencias. Este perfil permite presentar una guía electrónica de programas personalizada al usuario que se encuentre presente en cada momento; ya que esto involucra la detección automática y en tiempo real de usuarios. Así se evita la tarea de tener que explorar en profundidad la EPG por defecto entre las decenas de programas que podemos encontrarnos. Este sistema está compuesto por diferentes módulos, desde la detección de dispositivos por parte de un cliente Bluetooth ubicado en el *set-top-box*, el almacenamiento de las preferencias de los diferentes usuarios y entradas en el sistema, hasta la adaptación del contenido a los gustos del usuario.

Dentro del proyecto se analizan las tecnologías y herramientas que mejor se adapten a él. Entre ellas, destacan un servicio Web desarrollado en Java, que permite el registro de los distintos usuarios y su posterior modificación. De esta manera podemos recoger las preferencias y gustos para así conformar los diferentes perfiles de los miembros de la familia. Todas estas características, que servirán para construir una serie de filtros aplicables sobre una EPG, así como el emparejamiento entre la MAC (*Medium Access*

Control) del dispositivo registrado y el identificador de usuario que lo posee, quedaran guardadas para su posterior lectura.

Una vez los perfiles están definidos, se puede comenzar a hacer uso del sistema a través de un emulador de TV digital interactiva denominado XletView, el cual nos va a permitir la ejecución de aplicaciones en lenguaje Java llamadas Xlets. Gracias a esto, y basándonos en el flujo de vida definido para estas aplicaciones, vamos a integrar el módulo detector de presencia con la parte de la capa de presentación que nos proporciona el emulador para conseguir proyectar por pantalla el contenido de acuerdo a los miembros presentes en la sala.

Finalmente, dispondremos de otros módulos integrados en nuestro Xlet, como es el control de acceso a través de políticas gracias a XACML (*eXtensible Access Control Markup Language*), que restringe el uso de sistema de acuerdo a un límite de edad y a unas horas del día, muy útil cuando de niños se trata.

Abstract

Digital Television, or DTV, is the application of digital media technologies for the transmission of content through a conventional antenna. Applying these technologies is more likely to get, such as providing a greater number of channels or a higher quality picture and sound. Thus, it generates a series of new services, some of them following the evolution of the old already existing in analogue television, as is the case with traditional programming service that we offer on Teletext and its current successor, EPG or electronic program guide. This is an easy channel organization that offers a television distributor, allowing an exploration of all, assuming a great help to users. But still the traditional EPGs without providing an effective search, because they take too long in the viewer look from hundreds of your favourite options.

The project consists, therefore, to design and implement a customization of the current terrestrial TV, so that each user has a profile associated with which to store your ID, your tastes and preferences. This page allows you to present a customized electronic program guide the user to be present at all times, as this involves the automatic detection and real-time users. This eliminates the task of having to drill down into the EPG by default among dozens of programs that can be found. This system consists of different modules, from the detection device by a Bluetooth client located in the set-top-box, saving the preferences of different users and entered into the system to adapt the content to personal preferences.

Within the project analyzes the technologies and tools that best suit him. These projects include a Web service developed in Java, which allows the recording of individual users and further editing. Thus we may collect preferences and tastes and the different time profiles of family members. All these features, which serve to build a series of filters applied on an EPG, and the match between the MAC (Medium Access Control) registered device and the user ID that owns it, stay saved for later reading.

Once the profiles are defined, you can start using the system through an interactive digital TV emulator called XletView, which will allow us to run applications in Java Xlets calls. Because of this, and based on the flow of life defined for these applications,

we will integrate the detector module with the presence of the presentation layer that provides the emulator to get the screen to project the content according to members present in the room.

Finally, we will have other modules in our Xlet, such as access control through policies because XACML (eXtensible Access Control Markup Language), which restricts the use of the system according to an age limit and a few hours a day very useful when children are involved.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1. Introducción y objetivos

1.1 Introducción

Sin lugar a dudas, la televisión es, junto con la radio, el medio de difusión por excelencia en la sociedad. cotidiana y familiar, la televisión cambia ahora su trasfondo tecnológico y evoluciona, del sistema analógico que nos ha acompañado en España los últimos cincuenta años, al sistema digital, que, desde el 30 de Noviembre de 2005 da servicio en la gran mayoría del territorio nacional, para así dar alas a este medio, avanzando aún más a favor de la extensión de la Sociedad de la Información.

Y es que, tras este cambio fundamental en el formato, la televisión no sólo gana calidad en los aspectos de video y de audio, sino que permite la emisión de nuevos canales, la pluralidad de idiomas y otros servicios de valor añadido (incluso interactivos) que ofrecen al medio, la adaptación que la Sociedad de la Información espera de él.

Ante estos cambios que está sufriendo la televisión, los agentes involucrados en el sector están obligados a adaptar a la nueva situación su modelo de negocio. Las nuevas posibilidades que ofrece la televisión digital deben ser aprovechadas, ya que de lo contrario puede ocurrir como en anteriores ocasiones y terminar fracasando.

La digitalización de la televisión, además de incrementar la competencia entre los contenidos habituales (al incrementar el número de programas emitidos), ofertará un amplio y atractivo abanico de nuevos servicios de valor añadido, cuyo sello de identidad es la interactividad. Esta digitalización tiende a una futura convergencia e interconexión de otros dispositivos: del hogar, móviles, PDA's, ordenadores personales… para que se pueda operar cómodamente con cualquiera y cada uno de estos dispositivos tecnológicos como si fuesen un solo ente.

Actividades como el acceso al menú de programación de la emisora, realizar compras de productos (*t-commerce*), acceso a cuentas bancarias, participar en encuestas, votar, escoger un mejor ángulo de visión en transmisiones de eventos deportivos, ver nuevamente las películas y programas, seleccionar música para escuchar, enviar y recibir e-mails, etc., pasan a hacer parte del conjunto de recursos interactivos posibles.

El desarrollo de todas esas aplicaciones solo es posible gracias a una capa de software intermedio presente en la arquitectura de los padrones de los sistemas de televisión digital llamada *middleware*. El middleware, tiene como finalidad ofrecer un servicio personalizado para las aplicaciones, escondiendo las peculiaridades y heterogeneidades de las capas inferiores (tecnologías de compresión, de transporte y de modulación) viabilizan así el desarrollo de las aplicaciones de forma independiente del hardware de los fabricantes de los terminales de acceso a la señal digital (*set-top-box*).

1.2 Motivaciones

La televisión digital terrestre permite mejorar la calidad de la recepción y amplía la oferta disponible tanto en el número de canales como en la versatilidad del sistema: emisión con sonido multicanal, múltiples señales de audio, teletexto, EPG, imagen panorámica, servicios interactivos, etc.

Partiendo de una guía electrónica de programas por defecto, la cual podemos encontrar en cualquier televisión actual que incorpore un *set-top-box* tanto interno como externo, las motivaciones para desarrollar un sistema que facilite al usuario la creación de una EPG personalizada son muchas.

Cientos son los programas que podemos encontrar en la televisión a lo largo del día, y decenas los canales que los emiten, resultando casi imposible hacer un barrido de todos ellos en el momento de ponernos a ver la televisión. Es por ello que la implementación de un sistema que simplifique y filtre todos esos programas nos supone una gran ventaja a la hora de disfrutar de un rato agradable en frente de nuestra televisión.

Otra de las motivaciones para desarrollar este proyecto, es la posibilidad de acceder al sistema sin necesidad de interactuar con éste de manera directa. Tenido el sistema en funcionamiento, y con el simple hecho de acercarnos a la televisión, automáticamente se activará cargando nuestras preferencias establecidas en el primer uso.

1.3 Objetivos

El objetivo fundamental del proyecto es implementar un sistema capaz de personalizar una guía electrónica de programas según los gustos y preferencias de los distintos usuarios que hagan uso de él, que involucre la detección de presencia. Para ello, se establecen una serie de objetivos específicos con el fin de llevar a cabo esta tarea:

1. Analizar el sistema que se quiere desarrollar

   Se lleva a cabo un análisis de la funcionalidad y el diseño del sistema que se quiere implementar. Para ello, se va a realizar un esquema de los distintos módulos que conformaran el proyecto.

2. Estudiar las distintas tecnologías que permiten implementar la funcionalidad

   - Bluetooth: Esta tecnología inalámbrica nos ofrece comunicación entre dispositivos de bajo consumo y comunicaciones a corta distancia (10 metros), con un ancho de banda reducido de hasta 11 Mbps. Dispone de un API para Java dividida en dos partes: el paquete `javax.bluetooth` y el paquete `javax.obex`, totalmente independientes. Será el primero el que nos proporcione clases e interfaces básicas para el descubrimiento de dispositivos.

   - JavaEE: *Java Enterprise Edition*, tecnología que nos permitirá la ejecución en su servidor de aplicaciones de los *servlets* y *Java Server Pages* necesarias para el desarrollo de la parte de la interfaz web, necesaria para el registro de usuarios en el sistema.

   - XletView: herramienta de código libre, bajo la licencia de software libre GLP (*General Public License*), que permite la ejecución de Xlets, aplicaciones Java que proporcionan interactividad para la televisión Digital.

3. Implementar la aplicación Web y la aplicación Xlet.

   Implementación de las distintas clases e interfaces de acuerdo a la funcionalidad acordada, como la detección de usuarios, la adaptación del contenido por pantalla, la política de acceso al sistema o la lectura de ficheros en las distintas aplicaciones.

4. Integrar los distintos módulos

   Es decir, anexionar los módulos para que se ajusten a la lógica del sistema y por lo tanto funcionen como un solo bloque.

5. Probar el rendimiento y puesta a punto del sistema.

   Realización de pruebas el análisis de la viabilidad del sistema, así como su rendimiento en un entorno real, como puede ser el hogar de una familia.

6. Documentar el proyecto

   Recopilación de toda la documentación que ha sido necesaria para desarrollar el proyecto, así como las decisiones en cuando a diseño e implementación del sistema.

1.4 Contenido de la memoria

Como hemos visto, este capítulo contiene una breve introducción, las motivaciones, objetivos y el contenido de la memoria. Por tanto, el resto de la memoria se estructura de la siguiente manera:

- Bloque II: Estado del arte
  - Capítulo 2: Televisión digital terrestre

    En este capítulo realizaremos una pequeña introducción a la televisión digital terrestre y nos centraremos en el elemento clave sobre el que gira este proyecto, la guía electrónica de programas. También haremos un análisis de la emisión y recepción de la señal digital y cuáles son los componentes que la conforman.

  - Capítulo 3*: Set-Top-Box* y *middleware*

    En este capítulo trataremos de profundizar en conceptos clave para entender cómo funcionan y cuál es el ciclo de vida de la aplicación interactiva que estamos tratando desarrollar (Xlet). También llevaremos a cabo un estudio de la herramienta de trabajo, el emulador XletView.

- Capítulo 4: Bluetooth

  En este capítulo realizaremos un repaso a la tecnología Bluetooth y analizaremos la API con las clases y métodos que nos proporciona.

- Capítulo 5: XACML

  En este capítulo nos centraremos en la política de acceso que nos proporciona XACML y analizaremos los distintos parámetros y atributos de que consta.

- Bloque III: Descripción, implementación y pruebas de rendimiento del sistema

  - Capítulo 6:Descripción del sistema

    En este capítulo se define la arquitectura del sistema con sus respectivas capas, los requisitos, funcionalidades y la forma en que se pretenden llevar a cabo.

  - Capítulo 7: Implementación del sistema.

    Atendiendo a las diferentes capas definidas en la arquitectura del sistema, se describen cómo se han implementado los diferentes módulos atendiendo a la funcionalidad que requieren.

  - Capítulo 8: Pruebas de rendimiento

    En este capítulo se describen las pruebas realizadas, para determinar si el sistema es viable o no, así como llegar a una conclusión.

- Bloque IV: Historia del proyecto, conclusión y trabajos futuros

  - Capítulo 9: Historia del proyecto

    En este capítulo se realiza un análisis de las distintas etapas en las que el proyecto se ha dividido, con una breve descripción de la tarea realizada y una estimación del tiempo empleado.

- Capítulo 10: Conclusiones y trabajos futuros

  En este capítulo se analizan las los objetivos cumplidos con el proyecto, para elaborar una conclusión y poder así establecer unas futuras líneas de trabajo.

- Bloque V: Apéndices

  - Apéndice A: Presupuesto

  - Apéndice B: Instalación y configuración de herramientas

Bloque II

Capítulo 2. Televisión Digital Terrestre

2.1 TV digital

La televisión digital terrestre (TDT) es una técnica de difusión de televisión que sustituye a la televisión analógica digital, y basa su funcionamiento en la transmisión de imágenes en movimiento y su sonido asociado mediante una señal digital a través de una red de repetidores terrestres. La información digital es mucho más versátil que la analógica, porque sobre éstos podemos aplicar diferentes métodos algorítmicos, como por ejemplo, métodos de compresión que permiten enviar mayor cantidad de información que antes, con las ventajas que esto supone. Nos proporciona además la posibilidad de enviar información que no tiene por qué ser exclusivamente la señal digital, pudiendo transmitir aplicaciones que luego serán interpretadas y ejecutar en un dispositivo receptor.

La tecnología digital no es un método de transmisión realmente nuevo, sino que se viene desarrollando desde 1991, cuando se formo el consorcio DVB-EBU (*Digital Video Broadcasting-European Broadcasting Union*) por cadenas de televisión, programadores, fabricantes de aparatos electrónicos y diferentes entidades públicas. Entre sus objetivos principales está el desarrollo de la tecnología digital definiendo diferentes estándares que sean ampliamente aceptados como normas para la utilización de dicha tecnología. En España se utiliza el nombre de TDT para nombrar al estándar DVB-T, diseñado para las emisiones mediante la red de distribución terrestre de señal usada en la televisión analógica. Este estándar forma parte de toda una familia de estándares de la industria para la transmisión digital según diversas tecnologías: emisiones desde satélites geoestacionarios (DVB-S), por redes de cable (DVB-C), e incluso para emisiones destinadas a dispositivos móviles (DVB-H), como podemos observar en la figura 1. Entre los estándares fuera de Europa nos encontramos con el estadounidense ATSC (*Advanced Television Systems Committee*), el estándar japonés y de la mayoría de los países latinoamericanos ISDB-T (*Integrated Services Digital Broadcasting*) y el chino DTMB (*Digital Terrestrial Multimedia Broadcast*).

Estándar	Medio de difusión	Plataforma
DVB-S/S2	Satélite	TV equipada con STB
DVB-T	Terrestre	TV equipada con STB
DVB-C	Cable coaxial	TV equipada con STB
DVB-H	Terrestre	Teléfonos móviles
DVB-IP	Redes LAN/WAN	TV + STB / Ordenador

Figura 1: Principales estándares DVB [1]

2.1.1 Televisión digital frente a televisión analógica

En la transmisión digital, la codificación de la señal permite la compresión de la señal, haciéndose un uso más eficiente del espectro radioeléctrico, así como la posibilidad de emitir canales en alta definición (HD). Tras la multiplexación, se hace posible la emisión de más canales en el espacio antes empleado por uno (canal múltiple digital). Dependiendo del ratio de compresión, así será el número de programas por canal múltiple. Además, se puede hacer uso del espectro sobrante para otros usos.

Tanto la señal digital como la analógica poseen la misma naturaleza (señales electromagnéticas) y son susceptibles de ser distorsionadas e interferidas por distintos tipos de campos (eléctricos o magnéticos), condiciones meteorológicas, etc. No existe más distinción en ambas señales que la manera de codificar la información. La codificación digital sigue algoritmos que permiten la posterior identificación de los errores y su corrección.

La TDT nos permite un mayor aprovechamiento del ancho de banda que la televisión analógica [2]. Esto se debe a que la antigua televisión solo permitía la transmisión de un único programa por cada canal UHF (*Ultra-high frequency*), obligando al resto de canales adyacentes a permanecer libres para evitar interferencias. Ahora podemos transmitir varios programas en un solo canal UHF, dependiendo de la calidad de imagen y sonido deseados, como podemos observar en la figura 2. Al bloque de programas emitidos por UFH se le denomina MUX (múltiplex). El flujo binario del MUX es la multiplexación de los canales que lo componen. La relación de flujo de cada canal multiplexado es susceptible de ser regulada, lo que significa la regulación de la calidad de los mismos. Esto quiere decir que si necesitamos asignar un flujo alto a algún evento por motivos de calidad de emisión, podemos decrementar el flujo del resto de canales que componen el MUX. Cuando todos los canales tienen la misma importancia, el aprovechamiento óptimo del MUX se realiza mediante un control estadístico del flujo.

En cada momento, un sistema inteligente determina el flujo de cada canal del MUX y va asignando un mayor o menor ancho de banda según la necesidad detectada.

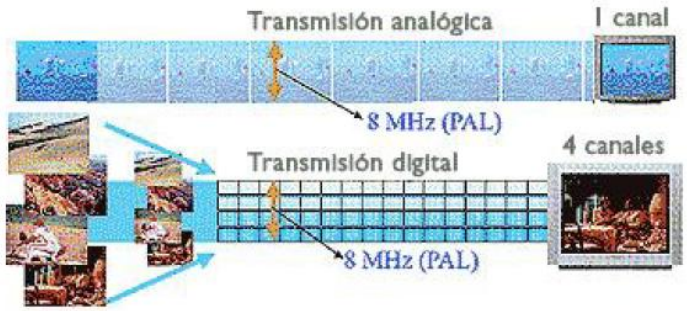

Figura 2: TV digital vs TV analógica [3]

Gracias al mayor aprovechamiento del ancho de banda, la TDT nos permite transmitir mayor cantidad de señales en un mismo canal, lo que se traduce también en un ahorro energético importante por canal, suponiendo una reducción de costos para los radiodifusores. En cuanto a la corrección de errores, al codificar la señal de manera digital y no proporcional, se pueden subsanar hasta cierto punto. Sin embargo, cuando el receptor no es capaz de resolver ciertos errores, se produce una congelación de partes de la imagen o la interrupción del sonido. Cuando este nivel de error supera cierto nivel, la pantalla ofrece una imagen en negro sin sonido, ya que el receptor es incapaz de recomponer la señal. El hecho de que exista este límite de error determinado y no una pérdida progresiva de la calidad (como pasaba en la transmisión analógica), se le denomina abismo digital.

En cada canal de radio se emite un único flujo MPEG-2 (*Moving Picture Experts Group -2*) que puede contener un numero arbitrario de flujos de audio, video o datos. Aunque varios operadores compartan el uso de un canal multiplexado, cada uno puede gestionar el ancho de banda que le corresponde para ofrecer los contenidos que desee. El aprovechamiento de toda esta información por parte del usuario es posible gracias a las diversas aplicaciones de que dispone el receptor TDT, conformes al estándar denominado *Multimedia Home Platform*, MHP. Cada operador podrá desarrollar las aplicaciones que proporcionen los servicios deseados a sus clientes, y éstas se instalarán en el receptor TDT para dar acceso a dichos servicios.

A diferencia de las limitaciones de los proveedores analógicos, la digitalización permite numerosos servicios que éstos no podían ofrecer, como son teletexto con un aspecto

gráfico de más calidad, subtítulos en varios idiomas y servicios interactivos que permiten al usuario, o al televidente, interactuar con el contenido del programa de televisión (por ejemplo, en un evento deportivo se podría mostrar estadísticas de los jugadores o participar en concursos de televisión). La interactividad se basa en la comunicación que permite el canal de retorno que poseen algunos receptores con el proveedor del servicio interactivo.

Las aplicaciones interactivas desarrolladas por cada operador para proporcionar determinados servicios a sus clientes son integradas e instaladas en el receptor TDT, permitiendo así el acceso a ellos. Una de estas aplicaciones es la EPG, que permite al usuario ver la información de todos los programas de los distintos canales, y según la complejidad del receptor la posibilidad de grabarlos, ver los actores de los mismos, etc.

2.2 Funcionamiento de la TDT

2.2.1 Emisión de la señal

La señal de televisión se transmite como un flujo de datos codificados en MPEG-2 (tanto para audio, video y datos) llamado flujo de transporte (*transport stream*) y denominado MPEG-2 TS. MPEG-2 es un estándar de codificación de audio y video para señales de transmisión, aunque también es el formato usado para codificar los discos SVCD (*Super Video CD*) y DVD (*Digital Video Disc*) comerciales de películas.

Para la transmisión de la señal, es necesario someterla a un proceso de modularización para poder transmitir los posibles estados (ceros y unos) de los que se compone la señal digital. Entre las distintas modulaciones usadas en la televisión digital encontramos QAM (*Quadrature amplitude modulation*), QPSK (*Quadrature phase-shift keying*) y COFDM (*coded Orthogonal frequency-division multiplexing*). Está ultima modulación es la usada en España para la transmisión, y se basa en dividir la información entre un numero de portadoras independientes que se suman de forma ortogonal y cada una de ellas modulada por la normal 8k (6817 portadoras). Para recibir la señal es necesario un sistema que sea capaz de decodificar esa señal para poder acceder al contenido ya sea un receptor o una tarjeta de televisión para ordenador que pueda interpretar dicha señal. En la figura 3 podemos ver como es el proceso de emisión y recepción de la señal.

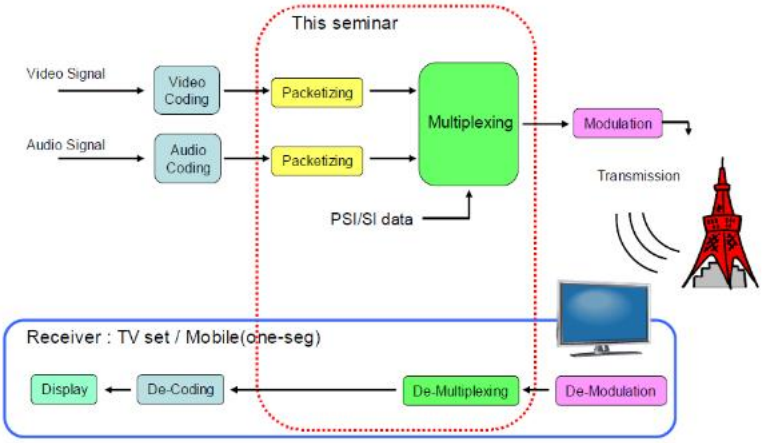

Figura 3: Transmisión y recepción de señal de televisión [4]

2.2.2 Elementos que componen la señal

Cada flujo de transporte se compone de otros sub- flujos denominados flujos elementales (*elementary streams*) y que pueden llevar audio y video codificado en MPEG-2 o datos encapsulados en un flujo MPEG-2, constando cada paquete de 188 bytes, incluyendo una cabecera de 4 bytes seguida de un campo de adaptación (usado eventualmente para rellenar el exceso de espacio disponible) y en cualquier caso de una carga útil o *payload*. El flujo de transporte está formado por muchos flujos elementales, por lo que cada uno tiene que llevar asociado un identificador único, denominado PID (*packet identifier*) y añadido en la cabecera del paquete, para su posterior identificación en la multiplicación en el flujo de transporte. En la figura 4 vemos como el PID está formado por 13 bits, y lo que limita el número de flujos elementales que pueden ir en un mismo flujo de transporte es la tasa binaria.

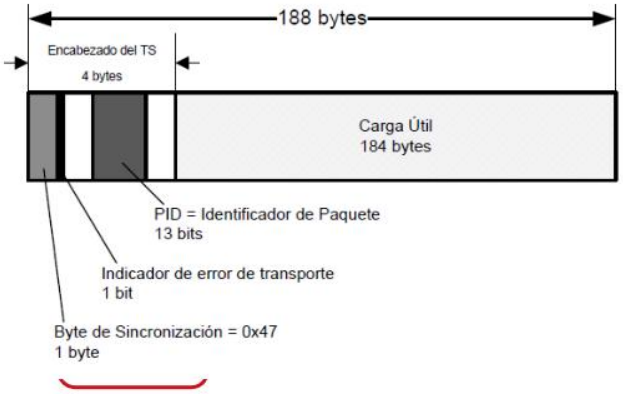

Figura 4: *Transport Stream* [4]

Dentro de un flujo de transporte se pueden transmitir varios servicios, cuyos componentes son multiplexados para ser transmitidos de manera conjunta con sus respectivos flujos elementales, formando un único flujo de paquetes de datos, véase en la figura 5. Los proveedores de televisión son los encargados de generar estos datos para ser transmitidos.

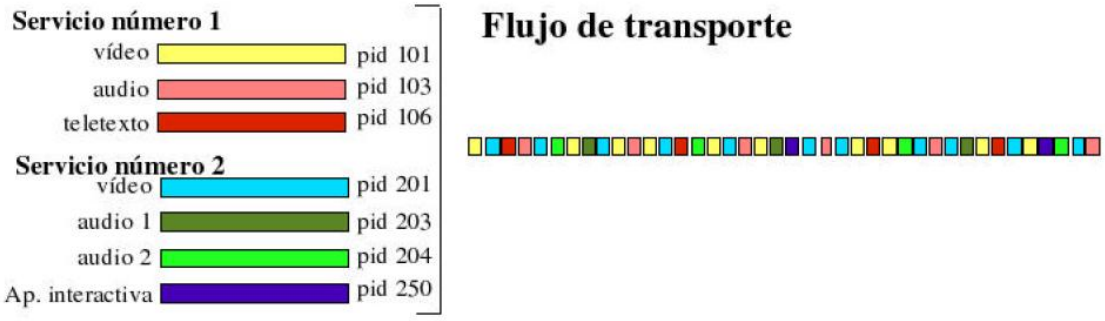

Figura 5: Flujo de transporte [5]

Para que el receptor pueda obtener la señal sin sobrecargar el buffer de decodificación MPEG-2, es preferible insertar los paquetes dentro del flujo de transporte en un orden, de manera que se entregue cada elemento con anticipación y a una velocidad lo suficiente constante. Es importante destacar que en el *transport stream*, además de los paquetes asociados a los flujos elementales de la señal, es necesario incorporar paquetes de transporte que contienen información sobre el servicio, así como paquetes de transporte nulos que se emplean para absorber eventuales reservas de capacidad del multiplex.

Pero cuando la señal es multiplexada y enviada a los receptores, éstos no reciben indicación alguna de que tipo de información están recibiendo o de cómo reconstruirla, por ejemplo, para mostrársela al usuario. Para solventar este problema, DVB especifica unas tablas denominadas tablas de información de servicio o *service information* (SI) que deben ser añadidas al flujo de transporte. Se trata de una sencilla base de datos que describe la estructura del flujo de transporte, los servicios que hay dentro de ellos y su contenido. A través del *DVBLocator* podremos seleccionar el servicio identificado una vez que dispongamos de toda la información sobre los servicios disponibles. El *DVBLocator* es un identificador único que permite localizar cualquier servicio DVB y se compone de tres elementos*: original network ID, transport stream ID y service ID*.

Las tablas SI son un conjunto de flujos elementales que contienen una serie de tablas que describen como están estructurados los datos transmitidos así como los servicios insertados en él e información útil al usuario y que podrá ser mostrada por los receptores digitales. También describen información sobre todos los flujos de transporte transmitidos a través de la red por la que se transmite el flujo, lo que supone que podamos obtener información de todos los servicios accesibles desde esa red. Para distinguir las tablas SI del resto, se hace uso de PID reservados. Entre las SI, podemos destacar:

- *Program Association Table* (PAT): tabla de inclusión obligatoria, transportada con PID=0x0000 y que contiene una lista completa de todos los programas disponibles en le transport stream. Cada programa aparece junto con el valor del PID de los paquetes que a su vez contienen la tabla con los datos que identifican a dicho programa. La PAT debe transmitirse sin cifrar aunque todos los demás programa lo estén.

- *Conditional Access Table* (CAT): tabla que debe estar presente si al menos un programa del multiplex es de acceso condicional. Se transporta con PID=0x0001 y proporciona detalles de los sistema de cifrado empleado, así como los valores de los PID de los paquetes de transporte que contienen la información del control de acceso condicional.

- *Program Map Table (*PMT): tabla que proporciona detalles acerca del programa y de los flujos elementales que comprende. Pueden ser transportadas por paquetes con valores de PID arbitrarios, exceptuando 0x0000 (PAT) y 0x0001 (CAT) y restringidos desde 0x0002 hasta 0x001F. En esta tabla, los datos PID no pueden estar cifrados, aunque si pueden contener información privada relativa al programa, que eventualmente sí puede estar cifrada (datos de control de acceso).

La información del servicio o *service information* incluye además, otros 4 tipos de tablas de inserción obligatoria dentro del *transport stream* y 3 tipos de tablas opcionales:

- *Network Information Table* (NIT): tabla que proporciona información acerca de la red física usada para transmitir el *transport stream*, la lista de servicios que contiene, y que es transportada por los paquetes con PID=0x0010. Podemos observar un ejemplo en la figura 6.

- *Service Description Table* (SDT): tabla que aporta más información sobre los servicios transmitidos, pero orientada más al usuario. A diferencia de la PMT, que hay una para cada servicio, la tabla SDT es única para todo el flujo de transporte. Se transporta por paquetes identificados con el PID=0x0011.

- *Event Information Table* (EIT): tabla utilizada para transmitir información relativa a los acontecimientos en curso o futuros en el multiplex MPEG recibido en la actualidad, y eventualmente sobre otros multiplex MPEG, como denominación, hora de comienzo, duración, etc. Se transporta por paquetes identificados con el PID=0x0012.

- *Time & Date Table* (TDT): tabla que proporciona información relativa a la hora y fecha del momento, y se utiliza para poner en hora el reloj interno dl receptor. Se transporta por paquetes identificados con el PID=0x0014.

Entre las opcionales, tenemos:

- *Bouquet Association Table* (BAT): tabla que proporciona información relativa a los "bouquets", entendidos como una colección de servicios comercializados como entidad única. Además de informar del nombre del *bouquet*, aportan una lista de los servicios disponibles en cada uno. Se transportan por paquetes identificados con el PID=0x011

- *Running Status Table* (RST): tabla que actualiza de forma rápida la información relativa a la situación de un acontecimiento (que este o no sucediendo). Se transmite una vez y no de forma repetitiva por los paquetes identificados mediante el PID=0x0013

- *Time Offset Table* (TOT): tabla que proporciona información relativa a la hora real así como a la diferencia horaria local. Se actualiza frecuentemente, siendo transmitida por paquetes con el PID=0x0014, al igual que las TDT.

- *Stuffing Table* (ST): tabla de relleno que se emplea para invalidar tablas que ya no sirven. Es por esto por lo que comparten valores de PID con otros tipos de tablas: 0x010, 0x0011, 0x0012, etc.

Cada tabla SI tiene una estructura muy similar, están formadas por un conjunto de bits, constando de una cabecera en la que se identifica el tipo de tabla de que se trata, seguida por uno o más bucles con descriptores, que son sub-tablas que contienen información. Cada uno de ellos consta de una etiqueta, *descriptor_tag*, que permite identificar el tipo de descriptor para el tratamiento de su información. Son genéricos y pueden estar incluidos en varias tablas.

Sintaxis	Nº Bits
network_information_section(){	
table_id	8
section_syntax_indicator	1
reserved_future_use	1
reserved	2
section_length	12
network_id	16
reserved	2
version_number	5
current_next_indicator	1
section_number	8
last_section_number	8
reserved_future_use	4
network_descriptors_length	12
for(i=0;i<N;i++){	
descriptor()	
}	
reserved_future_use	4
transport_stream_loop_length	12
for(i=0;i<N;i++){	
transport_stream_id	16
original_network_id	16
reserved_future_use	4
transport_descriptors_length	12
for(j=0;j<N;j++){	
descriptor()	
}	
}	
CRC_32	32
}	

Figura 6: *Network Information Section* [3]

Entre los datos más importantes de esta tabla NIT encontramos el identificador, *table_id*, cuyo valor es 0x40, y el *network_id*, el identificador de red física. En el primer bucle tenemos los descriptores de la red, y en el segundo los identificadores de los

flujos de transporte proporcionados para esa red. En el bucle mas interno nos encontramos los descriptores para cada uno de esos flujos de transporte.

2.2.3 Recepción de la señal

Al igual que la señal analógica, la señal digital se transmite por ondas electromagnéticas, por lo que seguirán sirviendo las mismas antenas y redes de distribución de señal que se usan en la transmisión analógica. Pero para la transformación de la señal digital en analógica, será necesaria la instalación de receptores de televisión digital en los hogares. Estos receptores son los que hacen posible la recepción de la señal digital y la sintonización del canal para la visualización a través de la televisión, además de permitirnos interactuar con las aplicaciones interactivas que ejecuta. Entre los tipos de receptores, podemos encontrar:

- **STB** (*set-top-box*): se trata de un receptor digital externo que se conecta al televisor y que posee un middleware integrado (MHP) que proporciona un API sobre el que se desarrollan las actividades interactivas. Permite demodular la señal digital y enviarla al televisor. Además dispone de un modem para actuar como cable de retorno.

- ***Zappers***: receptor digital externo para la sintonización de canales de televisión, que suele tener algunas aplicaciones básicas integradas, pero que no permite la ejecución de servicios interactivos la no disponer del soporte adecuado.

- **IDTV** (*Integrated Digital Television*): se trata de un receptor digital integrado dentro del televisor.

- **PVR** (*Personal Video Recorder*): dispositivo interactivo de grabación de televisión y video en formato digital, gracias a un disco duro de gran capacidad. Se puede considerar como un *set-top-box* más sofisticado y con capacidad de grabación.

En el ámbito de este proyecto, el STB nos ofrecerá una emulación de la recepción de una señal digital y su posterior salida en la televisión. Haremos uso de un emulador de la capa middleware de un *set-top-box*, denominado XletView, que nos permitirá hacer uso de código java para el manejo de todo el flujo de vida de un Xlet.

2.2.4 Guía electrónica de programas

La EPG, o guía electrónica de programas [6], representa la evolución del tradicional servicio de teletexto que la televisión analógica nos ofrecía. Se trata de una lista con todos los programas de los diferentes canales organizados de manera sencilla y rápida, pudiendo acceder a la información de cualquiera de ellos. De esta manera, el usuario puede seleccionar que desea ver sin tener que realizar un zapping por todos los canales. Además, la EPG nos permite filtrar canales a partir de sus diferentes temáticas, ya sea deportes, cine, entretenimiento o dibujos, entre otros. Uno de las aplicaciones más útiles es hace uso de la EPG para programar el PVR para grabar algún contenido audiovisual. Para llevar a cabo esto, el *set-top-box* debe comunicarse a través del puerto infrarrojos con el PVR. Un *set-top-box* es un dispositivo que recibe la señal analógica o digital y la decodifica para luego ser mostrada a través de la televisión.

La transmisión de la EPG se basa en el estándar de televisión DVB, y viene encapsulada en el *transport stream*, donde además de los paquetes correspondientes a los diferentes canales de televisión, se encuentran unos paquetes de datos, que contienen servicios de información de las diferentes emisiones. Estos datos se encuentran estructurados en tablas, y la EPG en concreto se encuentra recogida en la *Service Information Table* (DVB-SI), donde recoge además información de otras tablas incorporadas al *transport stream*, como son la Tabla de Información de Eventos, la Tabla de Descripción de Servicios y la Tabla del Estado de Ala. Es el *set-top-box* el que se encarga de decodificar estos datos para extraer la información.

La presentación en pantalla viene dada mediante un menú donde se estructuran las diferentes opciones que se ofrecen. Mediante el mando a distancia, el usuario puede acceder y navegar entre los distintos servicios, véase la figura 7. Dependiendo del operador que proporcione la guía electrónica de programas, el formato, los colores o la organización en general pueden variar, aunque siempre buscando la forma más fácil de presentarla para que si manejo resulte sencillo y eficaz.

Figura 7: *Guía electrónica de Programas (EPG)* [7]

En el desarrollo de software para EPGs, los fabricantes deben incluir funciones para hacer frente a los crecientes volúmenes de datos cada vez más complejos relacionados con la programación [8]. Estos datos incluyen descripciones de los programas, horarios, clasificaciones, información de configuración del usuario, tales como listas de canales favoritos y contenido multimedia. Para satisfacer esto, algunos programas para *set-top-box* han sido diseñados para incorporar una capa de base de datos que clasifica, almacena y recupera datos de programación. Aunque la mayoría de los detalles son invisibles para los usuarios, que simplemente tienen una guía de programas en su equipo que "simplemente funciona".

Las EPGs mostradas por defecto en los televisores actuales son el producto de la decodificación de la información del flujo MPEG-2 y de la extracción de la información de la tabla ETI. Sin embargo, existen múltiples maneras de actualizar esa guía, a través de los set-top-box y de los PVR, y que nos permiten una vez descargado de Internet el contenido y programación de un periodo determinado, instalarlo para que podamos acceder a él a través de la televisión.

El proyecto XMLTV consiste en un conjunto de *grabbers* (recopiladores de información) que generan un archivo XML, con el formato de la guía electrónica de programas (véase en la figura 8). Dicho archivo contiene información del contenido de la televisión, por título, horario, descripción e incluso una organización por distintos canales. Una vez generado el fichero, se puede utilizar en distintos tipos de programas instalados en nuestro PVR o en nuestro *set-top-box*.

```
<programme start="20080715023000 -0600" stop="20080715040000 -0600" channel="I10436.labs.zap2it.com">
  <title lang="en">Mystery!</title>
  <sub-title lang="en">Foyle's War, Series IV: Casualties of War</sub-title>
  <desc lang="en">The murder of a prominent scientist may have been due to a gambling debt.</desc>
  <date>20070708</date>
  <category lang="en">Anthology</category>
  <category lang="en">Mystery</category>
  <category lang="en">Series</category>
  <episode-num system="dd_progid">EP00003026.0666</episode-num>
  <episode-num system="onscreen">2706</episode-num>
  <audio>
    <stereo>stereo</stereo>
  </audio>
  <previously-shown start="20070708000000" />
  <subtitles type="teletext" />
</programme>
```

Figura 8: XMLTV *file format*

Capítulo 3. Set-top-box y middleware

3.1 Introducción

Como mencionamos anteriormente, para aprovechar las funcionalidades que nos ofrece el sistema digital, en cuanto a aplicaciones interactivas y canales, necesitamos el uso de un nuevo dispositivo denominado *set-top-box* para convertir la señal digital recibida del proveedor a audio/video que la TV analógica debe presentar. La arquitectura de un *set-top-box* [9] podemos dividirla en 6 capas, como se refleja en la figura 9:

1ª: compuesta por los servicios y contenidos que pueden ser producidos en una transmisión digital, como puede ser la EPG, servicios interactivos, etc.

2ª: compuesta por las aplicaciones, responsables por promover el tipo de servicio de la capa superior.

3ª: la denominada Middleware, y que tiene como objetivo realizar una interfaz entre el hardware del *set-top-box* y las aplicaciones.

4ª: formada por los componentes multimedia de decodificación y codificación, así como otros módulos multimedia.

5ª: compuesta por el sistema operacional, responsable del funcionamiento del hardware, proveyendo una capa de abstracción al hardware del *set-top-box*.

6ª: formada por el hardware del *set-top-box* (CPU, almacenamiento, decodificación, dispositivos de entrada/salida, etc.). La arquitectura del hardware puede ser dividida en 3 etapas: una primera de interfaces de sintonización de las señales, una segunda de interfaces de transporte de control de sistemas y una tercera etapa de decodificación de audio y video y salida de la señal para la TV.

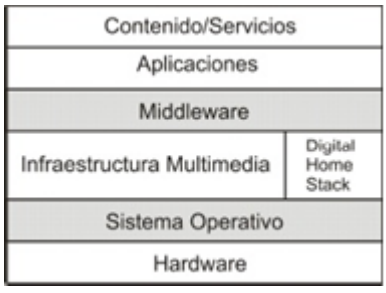

Figura 9: Arquitectura general de un *set-top-box* [9]

Los pasos que sigue un STB son los siguientes:

1. Lo primero que hace es sintonizar una señal digital, la cual incluirá tanto información de video (MPEG2, o MPEG4 para señales en alta definición), información de audio e información de datos (DVB-SI), etc.

2. El siguiente paso es separar los tres tipos de información que recibimos para tratarlos por separado.

3. A continuación, el sistema de acceso condicional decidirá cuales son los permisos que tiene el subscriptor para poder ver los contenidos que está recibiendo. Si tiene permiso descifrará esa información.

4. Una vez descifrados, los paquetes de video y audio son enviados al televisor.

5. Los paquetes de datos que hemos recibido junto con los de video y audio, se ejecutarán en caso de ser necesarios o solicitados por el consumidor.

6. El STB puede poseer un canal de retorno por donde enviar datos a la cabecera (*Back Channel*).

Entre los estándares de *middleware* se encuentran el europeo MHP de la DVB, el americano DASE (*DTV Application Software Enviroment*) de la ATSC (*Advanced Television Systems Committee*), el estándar ARIB de la ISDB-T de Japón, y el GINGA del estándar brasileño ISDB-Tb. Las arquitecturas de hardware de los diferentes estándares de televisión digital se diferencian principalmente por el tipo de demodulador y el decodificador MPEG-2/MPEG-4.

3.2 Multimedia Home Platform

Se trata de un *middleware* diseñado por el estándar DVB, que define una plataforma común para las aplicaciones interactivas de la televisión digital, independiente de hardware y software específicos, abiertos e interoperables para los receptores y decodificadores para TV digital. Entre los servicios que el estándar MHP nos puede ofrecer, encontramos la EPG, servicios de información de todo tipo, internet, comercio electrónico, *e-learning*, etc.

DVB-MHP hace uso del lenguaje de programación Java para las aplicaciones y defínela plataforma conocida como DVB-J, con un entorno de ejecución basado en el uso de una

maquina virtual de Java y un conjunto de interfaces de programación (APIs genéricas), situadas entre las aplicaciones y el sistema de software, para proporcionar a las distintas aplicaciones acceso a los recursos disponibles en el receptor. Además del uso del API Java, MHP introduce la posibilidad de utilizar un lenguaje de programación semejante a HTML, denominado DVB-HTML (*Digital Video Broadcast HyperText Markup Language*).

En cuanto a la arquitectura, queda definida en tres capas, como observamos en la figura 10:

- Una primera capa compuesta por los recursos: dispositivos de entrada/salida, memoria, procesador MPEG, etc.

- Una segunda capa denominada sistema de software, que proporciona el acceso de las aplicaciones a los recursos a través de ella, y que incluye un administrador de aplicaciones para el control de las aplicaciones que se ejecutan.

- La tercera capa está formada por las aplicaciones, también conocidas como Xlets, y que son recibidas a través del canal de *broadcast*, junto con las señales de audio y video convencionales.

Figura 10: Arquitectura MHP [10]

Según las capacidades del receptor, MHP define tres tipos de perfiles:

- Un primer perfil denominado *Enhanced Broadcast Profile*, que carece de canal de retorno, por lo que la interactividad queda limitada el mando a distancia y el televisor. Las aplicaciones son emitidas periódicamente y el espectador interactúa con la información que esta almacenada en su receptor. El usuario puede acceder a la información pero no puede enviar datos de vuelta desde su receptor. Algunos ejemplos serian el teletexto, la EPG e información de todo tipo.

- Un segundo perfil denominado *Interactive Broadcast Profile*, con canal de retorno, que proporciona una comunicación bidireccional con el proveedor de servicios interactivos. De esta manera, el consumidor puede enviar respuestas a través de la línea de teléfono o ADSL, entre otros. Chats, comercio electrónico o votaciones en concursos son ejemplos de este segundo tipo de perfil MHP.

- En este segundo tipo de perfil, podríamos hacer dos tipos de distinciones. Por un lado los servicios interactivos permanentes, y por otro los servicios interactivos sincronizados. En el primer caso, los telespectadores podrían encontrarlos de manera continua en la emisión, pudiendo interactuar con ellos independientemente de la emisión, mientras que en el segundo, van unidos a determinados programas o a sus contenidos.

- Un tercer perfil, *Internet Access Profile*, que añade las características de los dos anteriores y además proporciona acceso a internet.

La arquitectura de las plataformas MHP se completaría con la capacidad de admitir *plug-in*, lo que aporta gran flexibilidad a la misma. Mediante éste concepto, se consigue que un amplio espectro de aplicaciones que han sido desarrolladas hasta la fecha sobre otras plataformas puedan llegar a funcionar en una plataforma MHP, y poder así incluir en una plataforma ciertas funcionalidades que otros proveedores no ofrecen presentando de esta forma un elemento diferenciador con el cual competir.

El modelo grafico MHP distingue tres capas de gráficos:

- Una primera capa, denominada *background*, se muestra un color básico o una imagen.

- Una segunda capa situada por encima de la anterior, denominada video, que muestra videos con grandes limitaciones impuestas por parte de los receptores (sólo muestran el video a pantalla completa, a un cuadro de pantalla de resolución y en un limitado conjunto de coordenadas).

- Una tercera capa, denominada capa de gráficos, sobre a que trabaja MHP. Puede tener distinta resolución que las anteriores e incluso diferente forma en los pixeles.

Cada aplicación va a estar compuesta por varias capas de video y de gráficos por una de *background*. Aunque ciertas características en la configuración de las capas pueden crear restricciones en las demás, todas deben ser configuradas por separado.

Una de las grandes limitaciones de la parte grafica de MHP se encuentra en la las diferentes maneras de mostrar los componentes gráficos por parte de los receptores. MHP usa el sistema de color RGB, a diferencia de los sistemas de televisión, que usan el sistema YUV, y dado que MHP sólo exige a los receptores un soporte CLUT de 256 colores, las diferencias entre los distintos receptores pueden ser considerables a la hora de mostrar la aplicación.

Otra dificultad que se aborda es la presentación del texto debido a las limitaciones de los receptores, lo cual origina tres problemas. El primero de ellos se basa en la lectura del texto en la pantalla del televisor, para lo cual MHP impone el uso de Tiresias a los receptores, fuente diseñada para este propósito. El segundo problema se centra en la navegación, que siendo sencilla en un ordenador debido al fácil uso del ratón y de los *scrollbars*, se torna difícil en los receptores, debido a que el uso de saltos puede ser una solución para la navegación vertical pero resulta especialmente incomodo en la navegación horizontal. Además, debido a las formas de interpretar el texto y los componentes gráficos es distinta en los receptores, prever este problema se vuelve complicado. El tercer problema está relacionado con la disposición del texto, siendo el más importante. Como hemos visto, las distintas maneras de disponer los componentes gráficos y el texto obligan a definir un modo de disponer el texto en el caso de no ajustarse al espacio disponible, por lo que surge la necesidad de tratar esa disposición a bajo nivel. HAVi (*Home Audio Video Interoperability*) ofrece una posible solución que consisten en sustituir la parte del texto que desborde del espacio disponible por puntos suspensivos.

DVB tomo lo aportado por HAVi y añadió controladores adicionales, que proporcionaban una mayor flexibilidad gracias a la capacidad de controlar márgenes o la alineación vertical u horizontal. Pero MHP da un paso más, y define mecanismos adicionales que permiten por ejemplo modificar el color del texto, y permitir de esta forma presentarlo en distintas zonas.

Todos estos problemas son más fáciles de solucionar en aplicaciones DVB-HTML, ya que el lenguaje HTML tiene fuertes mecanismos para la regular la presentación del texto, aunque estas aplicaciones no están disponibles. De cara a las aplicaciones DVB-J se podría recurrir a browser escritos en Java, y de esta forma aprovecharse de las ventajas de HTML, pero a costa de incremental la complejidad de la aplicación.

3.2.1 Canal de retorno

Como hemos visto, dos de los perfiles de MHP están provistos de canal de retorno, lo que aumenta la interactividad de las aplicaciones. El medio de comunicación a través de ese canal de retorno se basa en el uso del protocolo TCP/IP (*Transmission Control Protocol/Internet Protocol*), y permite el uso de cualquier tecnología para implementar este canal, ya sea ADSL, GPRS, modem, etc. Esto proporciona al desarrollador una gran libertad, puesto que puede desarrollar la aplicación con independencia de la tecnología que el receptor vaya a usar para implementar el canal de retorno. Además esto permite el uso de futuras nuevas tecnologías de acceso a internet que se vayan desarrollando próximamente.

MHP obliga a los receptores cuyo perfil este provisto de canal de retorno a tener soporte HTTP 1.0, HTTPS y DNS. El resto de protocolos son opcionales, por lo que no se hace aconsejable su uso al no estar asegurada la disponibilidad. Es por esto que los desarrolladores tienes que encargarse de definir los protocolos necesarios para comunicarse con el receptor remoto.

El uso del canal de retorno en MHP muy parecido al uso en Java, con la salvedad de que en MHP la conexión no se asume siempre disponible, ya que al desconocer la tecnología usada para implementar el canal de retorno, existe la posibilidad de que sea por ejemplo a través de un modem, lo que no asegura la conexión permanente.

3.3 Middleware Ginga

Se trata de un *middleware* propio del estándar brasileño de televisión SBTVD (Sistema Brasileño de Televisión Digital), una capa de software intermedio, como apreciamos en la figura 11, que permite el desarrollo de aplicaciones interactivas para televisión, independientemente de la plataforma de hardware de fabricantes de terminales de acceso. Puede dividirse en un conjunto de aplicaciones declarativas y aplicaciones procedimentales. Para ello, Ginga se divide en dos grandes subsistemas interconectados, Ginga-J (orientado a aplicaciones procedimentales Java) y Ginga-NCL (para aplicaciones declarativas NCL, *Nested Context Language*). Según la funcionalidad del diseño de cada aplicaciones, se escogerá entre una forma de programar y otra.

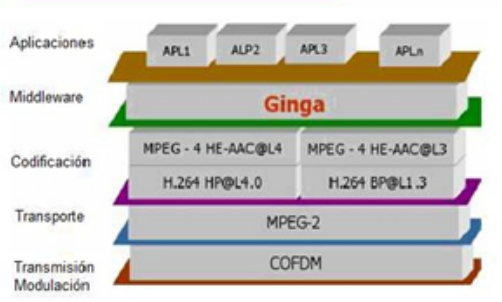

Figura 11: Arquitectura en capas SBTVD [9]

Las aplicaciones declarativas hacen uso frecuente de *scripts*, cuya naturaleza es procedimental. Pero por otra parte, una aplicación declarativa puede hacer referencia a una aplicación procedimental, y de la misma forma, una procedimental hacer referencia a una declarativa. Es por ello que ambos tipos de aplicaciones pueden utilizar los beneficios que brindan los ambientes para las aplicaciones declarativas y procedimentales. Existen tres módulos en los que la implementación de la arquitectura Ginga puede ser dividida: Ginga-CC (*Common Core*), Ginga-J (procedimental) y Ginga-NCL (declarativa).

3.4 Ginga-J

Ginga-J fue desarrollado para proveer una infraestructura de ejecución de aplicaciones basadas en lenguaje Java, denominadas Xlets, y que presentan facilidades específicas del ambiente de la televisión digital. Estas no necesitan ser previamente almacenadas en el STB, pues se pueden enviar por el canal de distribución.

3.4.1 Xlet

El interfaz Xlet [11] permite que una fuente externa (gestor de aplicaciones de un receptor digital) controle la ejecución de las aplicaciones. Este interfaz se encuentra en el paquete de aplicaciones `javax.tv.xlet.package`, y dispone de métodos que controlan los diferentes estados de vida, como podemos observar en la figura 12:

- **Cargado**: se carga la clase *main* del Xlet y se crea una instancia mediante el constructor por defecto. En ese momento le Xlet llega al estado *cargado*.

- **Pausado**: cuando se decide comenzar la ejecución del Xlet, se llama la método `initXlet()`. Una vez finalizada la inicialización, pasa a estado *pausado* hasta que haya suficientes recursos libres y pueda comenzar su ejecución. También se puede pasar a este estado si tras estar en *iniciado*, se llama al método `pausedXlet()`.

- **Iniciado**: se llama al método `startXlet()` y se pasa al estado *iniciado*, comenzando la ejecución del Xlet.

- **Destruido**: se pasa al estado *destruido* tras invocar el método `destroyXlet()` y se liberan todos los recursos del sistema ocupados por el Xlet.

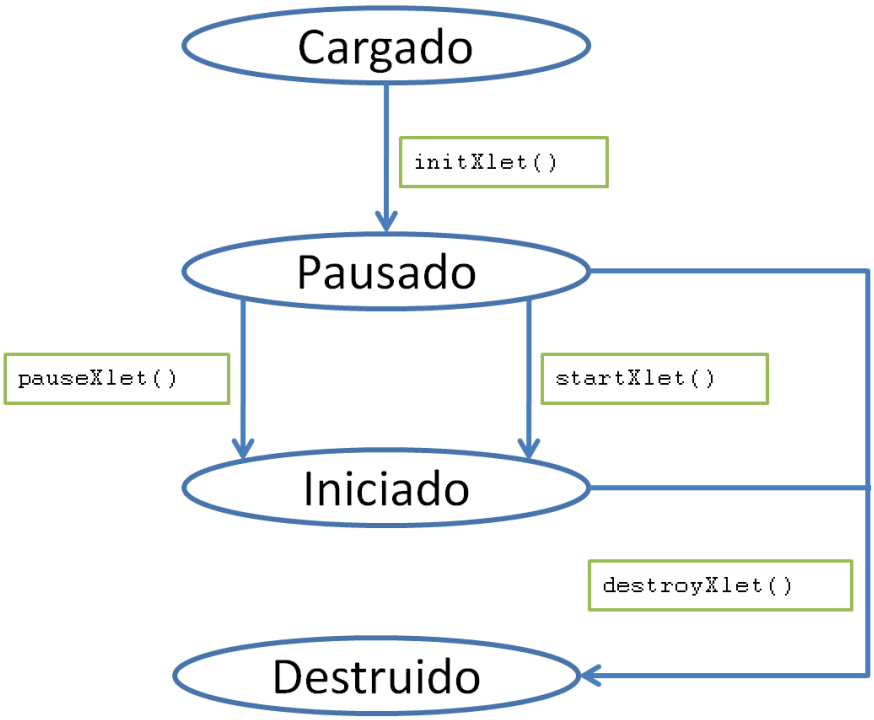

Figura 12: Ciclo de vida de un Xlet [12]

El gestor de aplicaciones es el encargado de cargar el archivo con la clase principal de Xlet y crear una instancia del Xlet llamando al constructor por defecto, momento en el que le Xlet alcanza el estado de loaded. Existen tres formas de inicializar un Xlet: invocado por otro Xlet, indicado directamente por el usuario y de manera automática por configuración de la tabla AIT. Es entonces cuando el gestor en el receptor invoca el método `initXlet`(), pasando un nuevo objeto `XletContext` para el Xlet. Si se usa este `XletContext` para inicializarse, se pasa al estado *pausado*, y podrá comenzar su

ejecución, invocando el método `startXlet`(). A partir de aquí el Xlet pasara al estado *iniciado* y podrá interactuar con el usuario.

Durante la ejecución del Xlet, el gestor de aplicaciones puede llamar al método `pauseXlet`(), lo que supone el paso del estado *iniciado* a *pausado*. La aplicación puede volver a ejecutarse de nuevo llamando al método `startXlet`() varias veces. Para llevar a cabo la finalización de la aplicación, el gestor invocara el método `destroyXlet`(), que supondrá el paso al estado *destruido*, y la liberación de todos los recursos ocupados por el Xlet.

Cada Xlet tiene asociado un contexto, encargado de guardar información adicional sobre el contexto y de notificar los cambios que se realizan en él. Si bien es verdad que los Xlets presentan muchas semejanzas con los Applets de Java, las diferencias se centran en el control del ciclo de vida y en la simplicidad de los primeros.

El entorno de ejecución Ginga-J establece un conjunto de APIs, divididas en tres partes. Una primera orientada a las innovaciones que dan soporte de las aplicaciones brasileñas, una segunda también orientada a las innovaciones, pero que pueden ser exportadas para otros sistemas. Por último la tercera API sigue el núcleo común del patrón GEM, basado en el *middleware* MHP y que define a su vez una serie de APIs usadas para el desarrollo de aplicaciones para la televisión digital, incluyendo las APIs provenientes de los paquetes de Sun Java TV, DAVIC (*Digital Audio Video Council*) e HAVi.

3.4.2 Java TV

Java TV [13] es un API de Sun Microsystems orientado principalmente al soporte de plataformas digitales de televisión, reuniendo una serie de elementos comunes necesarios en cualquiera de estas plataformas. Algunas de las características que la API de Java TV proporciona son: acceso a la base de datos del servicio de información, selección de contenidos, TV específica para el control del reproductor multimedia y el acceso a los datos que se emiten con la señal de televisión. Java TV soporta un alto nivel de interactividad, de calidad grafica y de procesamiento para poder ser ejecutado dentro de un *set-top-box*, siempre y cuando este equipada con la maquina virtual Java. Pero debido a su falta de concreción, Java TV es usado como un complemento a estándares como MHP y OCAP (*OpenCable Application Platform*), que se encargan de definir los aspectos concretos que este API (*Application Programming Interface*) deja sin definir.

Una de las principales aportaciones de JavaTV es el modelo de aplicación, donde la aplicaciones tienen su propia maquina virtual y por lo tanto un control absoluto sobre su ciclo de vida. Pero este modelo no es válido para el entorno de la TV digital debido al funcionamiento de los receptores. Es por ello que para solventar este problema, JavaTV define un nuevo modelo, donde todas las aplicaciones (Xlets) tienen una maquina virtual Java en común y al tener que compartir recursos, no tienen control total sobre su ciclo de vida, por lo que son las aplicaciones externas quienes las controlan.

Entre las funcionalidades que aporta la arquitectura de la API Java TV, encontramos:

- Acceso a datos en el canal de transmisión: Java TV puede recibir datos para las aplicaciones.

- Aplicaciones con interactividad: las aplicaciones usadas por esta API pueden procesar datos y devolverlos a través del canal de retorno.

- *Streaming* de audio y video, además de la posibilidad de generar otras aplicaciones con otros flujos.

- Gestión del ciclo de vida de las aplicaciones: permitiendo que las aplicaciones coexistan con el contenido de TV convencional y que permite el intercambio de canal sin que la aplicación deje de existir.

En la tabla 1 podemos observar una serie de librerías que forman el API, todos ellas pertenecientes a javax.tv, entre las que destacan:

Paquete	**Descripción**
`javax.tv.xlet`	Proporciona interfaces para el desarrollo de aplicaciones y la comunicación entre éstas y el administrador.
`javax.tv.locator`	Proporciona una forma para referenciar datos en programas accesibles por la API Java TV.
`javax.tv.net`	Permite el acceso a datagramas IP transmitidos en un flujo de difusión.
`javax.tv.graphics`	Permite que los Xlets puedan obtener su repositorio principal.

`javax.tv.util`	Soporta la creación y gestión de eventos del temporizador.
`javax.tv.media`	Define una extensión para JMF, con la finalidad de gestionar los medios de comunicación en tiempo real.
`javax.tv.media.protocol`	Proporciona acceso a un flujo de datos transmitido genérico.
`javax.tv.service`	Proporciona mecanismos para acceder a la base de datos.
`javax.tv.service.guide`	Da soporte a las aplicaciones de tipo EPG, incluyendo las descripciones individuales de los eventos, los horarios de la programación e información para el control parental.
`javax.tv.service.navigation`	Da soporte a los servicios de navegación.
`javax.tv.service.transport`	Define los mecanismos de transporte usados por los servicios de la televisión digital.
`javax.tv.service.selection`	Define como se presentan los servicios al usuario, así como la manera de seleccionarlos.

Tabla 1: Librerías de `javax.tv`

A la hora de decodificar y mostrar los contenidos de audio y video tanto Java TV como MHP recurren a JMF, y el uso que hacen es muy similar al de las aplicaciones Java, aunque con las limitaciones propias de la televisión digital. *Java Media Framework* permite la programación de tareas multimedia y presenta tres conceptos fundamentales:

- *Player*: elemento encargado de decodificar y reproducir el archivo media. Cada *player* tiene asociado un elemento *data source*.

- Control: elemento que aparece asociado a un *player*, y que aporta funcionalidades extra tales como el congelado de la imagen o la selección del lenguaje.

- *Data source*: elemento encargado de proporcionar al *player* los datos que decodificará y reproducirá.

Otro de los aspectos en los que interviene JavaTV es en la selección de un nuevo servicio. JMF nos proporciona la opción de cambiar de flujo de video o de audio que

está siendo mostrado en la pantalla, sin cambiar de servicio. Pero JavaTV va más allá. Selección de servicio es un término técnico de la televisión digital que se refiere al cambio de canal en un sistema de televisión digital. Cada servicio tiene asociado un contexto que contiene tanto la información de los Xlets que van asociados al servicio como la información referente al audio/video. Desde el contexto de un servicio se puede seleccionar un nuevo servicio, así como acceder al audio y al video. Pero el cambio de servicio puede ocasionar la finalización de las aplicaciones que estén en ejecución, ya que estas aplicaciones solo se ejecutan dentro de los servicios (uno o varios) en las que sean señalizadas.

Otra aportación de Java TV es la definición de un API que proporciona acceso a la información de servicio (ubicada en la tablas SI) que se señaliza en los servicios junto al audio, video y aplicaciones.

3.5 Ambientes de emulación para la programación Ginga

Entre las características que configuran el desarrollo de aplicaciones podemos encontrar desde los niveles de interactividad de la aplicación hasta el tipo de desarrollador y el ambiente de ejecución. La primera de ellas oscila entre la interactividad remota y la interactividad local, véase en la figura 13, mientras que el tipo de desarrollador puede ser una emisora, donde la aplicación será enviada al *set-top-box* a través de un canal de transmisión, o un usuario, en cuyo caso será enviada a través de una entrada externa, como puede ser un USB (*Universal Serial Bus*) o una tarjeta de memoria. Tras haber definido estos primeros parámetros, se procede con el ambiente de ejecución, el cual puede variar entre Ginga-NCL para aplicaciones declarativas y Ginga-J para las no declarativas.

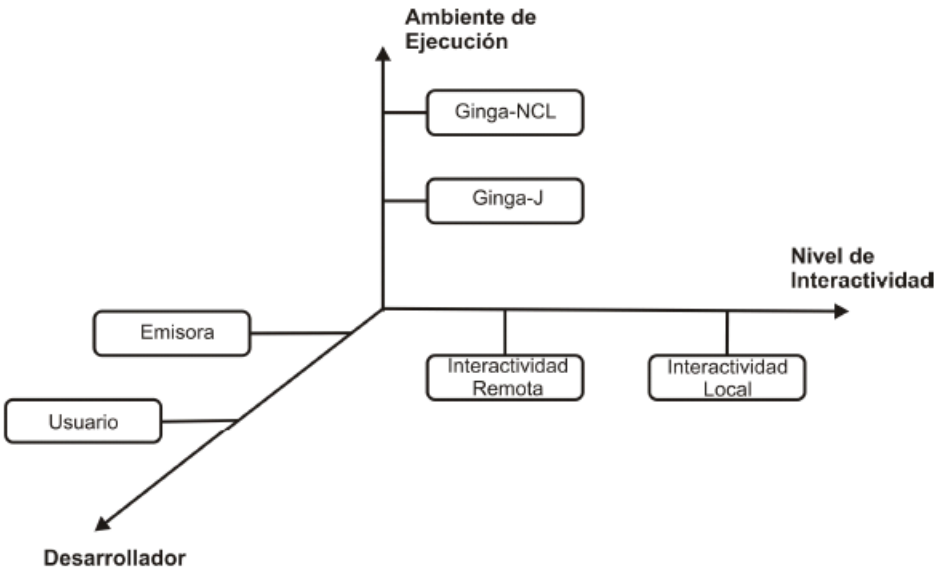

Figura 13: Ambiente de ejecución Vs Desarrollador Vs Interactividad [9]

La programación declarativa es una forma de programación que implica la descripción de un problema dado en lugar de proveer una solución para dicho problema, dejando la interpretación de los pasos específicos para llegar a dicha solución a un intérprete no especificado. El lenguaje de este tipo de programación está orientado a buscar la solución del problema, sin preocuparse de la forma de llegar a ello. Entre las ventajas que nos puede ofrecer este tipo de lenguaje destaca que la solución de un problema se puede realizar con un nivel de abstracción considerablemente alto, sin entrar en detalles de implementación relevantes, lo que hace a las soluciones más fácil de entender por las personas. Por el contrario, la programación declarativa no puede resolver cualquier problema dado, sino que está restringida al subconjunto de problemas para los que el intérprete o compilador fue diseñado.

En la programación no declarativa, se debe informar cada paso a ser ejecutado. Se puede afirmar que el programador posee un nivel alto de conocimiento sobre el código, siendo capaz de establecer todo el flujo de control y ejecución de su programa.

Una vez definidos todos los parámetros, se pueden definir el resto de pasos a realizar para la creación de aplicaciones.

Como podemos observar en la figura 14, en el ambiente de desarrollo de Ginga-NCL, podemos crear objetos que serán exhibidos a través del formateador NCL, llamados "Nodo de contenido", y que pueden ser objetos Media (hacen referencia a un elemento de comunicación como audio, video, etc.), objetos XHTML (hacen referencia a un

objeto XHTML) u objetos Lua (referencian a un elemento para dar soporte a la programación no declarativa utilizando scripts Lua).

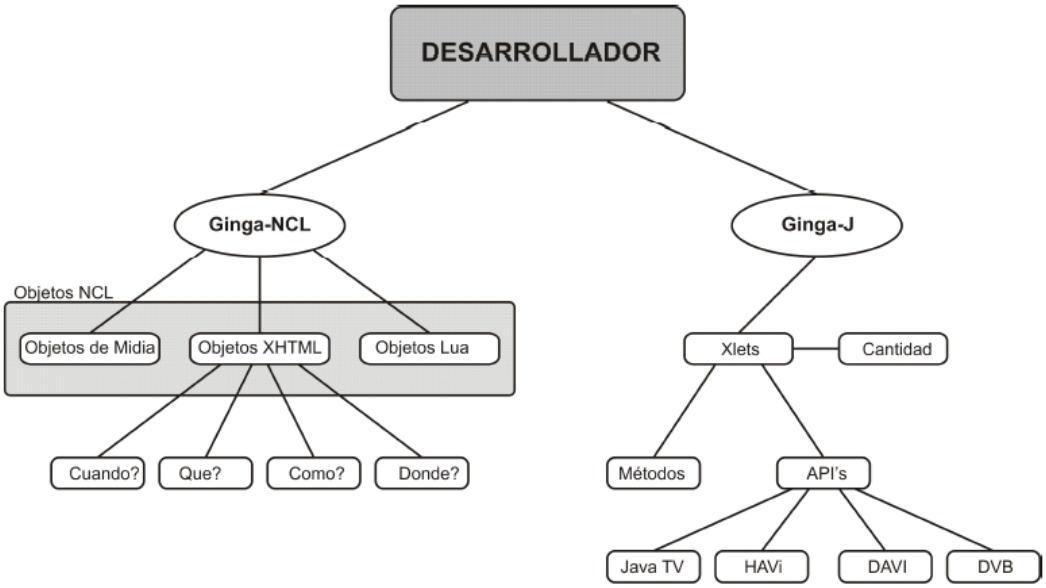

Figura 14: Opciones de desarrollo [9]

En el ambiente Ginga-J, que será el que vayamos a utilizar nosotros, se define una cantidad de Xlets necesarios para cubrir las necesidades de la aplicación, por lo que es recomendable la creación de un Xlet por cada función específica. Si por ejemplo tenemos una aplicación interactiva que muestra cuatro retransmisiones de los distintos partidos de la NBA que se están jugando en el momento, como en la figura 15, tendremos que disponer de cuatro Xlets, implementados de acuerdo a un API escogido, con una funcionalidad específica según las necesidades establecidas.

Figura 15: Aplicación mostrando varios Xlets

En la mayoría de los casos para el desarrollo de las aplicaciones, los desarrolladores no cuentan con una red de televisión experimental, por lo que tienen que recurrir a estaciones de pruebas o software de emulación. En el caso de los emuladores, existen varios que simulan el papel del middleware, siendo XletView el más utilizado para el ambiente Ginga-J y *Virtual Set Top Box* para el ambiente Ginga-NCL.

3.6 Emulador Ginga-J: XletView

XletView es un emulador usado para ejecutar Xlets en un ordenador, de código abierto, y bajo licencia de software libre GPL (*General Public License*). Además de tener una implementación de referencia a la API JavaTV, trae consigo implementaciones de otras APIs especificadas en el estándar MHP, como HAVi, DAVIC e implementaciones especificadas por la propia DVB (*Digital Video Broadcasting*), además de las bibliotecas de Personal Java.

XletView está desarrollado en Java y para su ejecución independientemente del sistema operativo, es necesario utilizar el *Java 2 Standard Development Kit*, que permite compilar Xlets y ejecutar el emulador. XletView utiliza la JMF (*Java Media Framework*) 2.1.1, pero con algunas deficiencias, como la incapacidad de exhibir video MPEG (*Moving Picture* Grupo de expertos) relacionados o controlados por un Xlet.

Un Xlet, es una aplicación Java que proporciona interactividad para la TV digital. Es similar a un Applet, que es adicionado en paginas HTML, aunque el Xlet es incluido en un servicio de TV digital. El *middleware* identifica un puente de entrada a la aplicación y la ejecuta utilizando la maquina virtual de Java. En la figura 16 podemos observar como es el interfaz del emulador.

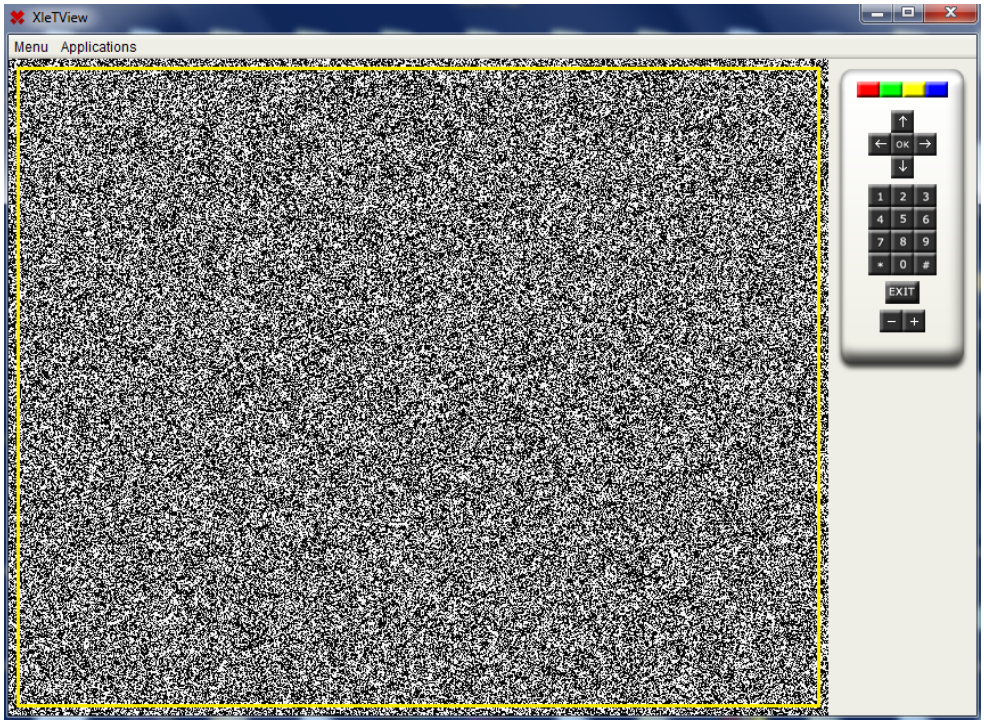

Figura 16: Interfaz XletView [9]

Capítulo 4. Bluetooth

Se trata de una tecnología de comunicación inalámbrica omnidireccional. Se ideó pensando en dispositivos de bajo consumo y comunicaciones a corta distancia (10 metros), con un ancho de banda reducida: hasta 11 Mbps. Es ideal para periféricos de ordenador (ratón, teclado, manos libres,...) y dispositivos móviles (teléfonos móviles, PDAs, Pocket PCs,...). Las principales aplicaciones de Bluetooth [14] son la transferencia de archivos, la sincronización de dispositivos y conectividad de periféricos, mientras que los principales objetivos que se pretende conseguir con esta tecnología son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre nuestros equipos personales.

4.1 JSR-82

Las APIs de Java permiten, entre otras cosas, diseñas aplicaciones independientes del hardware, el sistema operativo o el tipo de dispositivo empleado. Además, al ser un lenguaje de alto nivel orientado a objetos, permite una gran capacidad de abstracción a la hora de modelar aplicaciones. Por estas razones, se desarrollaron unos estándares API para Bluetooth utilizando el lenguaje de programación Java, conocidos como JABWT (*Java APIs for Bluetooth Wireless Technology*). En Java, todas las configuraciones, perfiles y paquetes opcionales se definen como JSR (*Java Specification Request*). En el caso de JABWT, este se definió como el estándar JSR-82 [15].

Este estándar permite esconder la complejidad y los detalles de bajo nivel del estándar Bluetooth y centrarse en el desarrollo rápido y sencillo de aplicaciones. Las capacidades que JSR-82 ofrece son las siguientes:

- Registro de servicios.
- Descubrimiento de dispositivos y servicios.
- Establecer conexiones RFCOMM, L2CAP y OBEX entre dispositivos para el envío de datos. La comunicación de voz, no obstante, no está soportada.
- Establecer seguridad en las comunicaciones, mediante autenticación y cifrado de datos.

Las APIs Java para Bluetooth [16] definen dos paquetes dentro del paquete javax.microedition.io: `javax.bluetooth` y `javax.obex`.

4.2 El paquete javax.bluetooth

En una comunicación Bluetooth existe un dispositivo que ofrece un servicio (servidor) y otros dispositivos acceden a él (clientes). Dependiendo de qué parte de la comunicación debamos programar deberemos realizar una serie de acciones diferentes.

Un cliente Bluetooth deberá realizar las siguientes:

- Búsqueda de dispositivos. La aplicación realizará una búsqueda de los dispositivos Bluetooth a su alcance que estén en modo conectable.

- Búsqueda de servicios. La aplicación realizará una búsqueda de servicios por cada dispositivo.

- Establecimiento de la conexión. Una vez encontrado un dispositivo que ofrece el servicio deseado nos conectaremos a él.

- Comunicación. Ya establecida la conexión podremos leer y escribir en ella.

Por otro lado, un servidor Bluetooth deberá hacer las siguientes operaciones:

- Crear una conexión servidora

- Especificar los atributos de servicio

- Abrir las conexiones clientes

4.2.1 Clases básicas

- Clase `javax.bluetooth.LocalDevice`

Un objeto `LocalDevice` representa al dispositivo local. Este objeto será el punto de partida de prácticamente cualquier operación que vayamos a llevar a cabo en este API. Alguna información de interés que podemos obtener a través de este objeto es, por ejemplo, la dirección Bluetooth de nuestro dispositivo, el apodo o "*friendly-name*" (también llamado "*Bluetooth device name*" o "*user-friendly name*"). A través de este objeto también podemos obtener y establecer el modo de conectividad: la forma en que nuestro dispositivo está o no visible para otros dispositivos. Para obtener la única instancia existente de esta clase llamaremos al método `getLocalDevice`() de la clase `LocalDevice`.

- Clase `javax.bluetooth.DeviceClass`

En el ejemplo hemos usado el método `getDeviceClass`() que devuelve un objeto de tipo `DeviceClass`. Este tipo de objeto describe el tipo de dispositivo. A través de sus métodos podremos saber, por ejemplo, si se trata de un teléfono, de un ordenador,... En la documentación del API hay una pequeña tabla con los posibles valores que devuelven los métodos de esta clase y su significado.

- Clase `javax.bluetooth.UUID`

La clase UUID (*Universal Unique Identifier*) representa identificadores únicos universales. Se trata de enteros de 128 bits que identifican protocolos y servicios. Como veremos más adelante un dispositivo puede ofrecer varios servicios. Los UUID servirán para identificarlos. En la documentación de la clase UUID se puede encontrar una tabla con los protocolos y servicios más comunes y sus UUIDs asignadas.

4.2.2 Descubrimiento de dispositivos y servicios

Las búsquedas de dispositivos y servicios son tareas que solo realizarán los dispositivos clientes. Una aplicación puede obtener una lista de dispositivos usando la función `startInquiry`() (no bloqueante) o `retrieveDevices`() (bloqueante). La primera requiere que la aplicación especifique un *listener*; este *listener* es avisado cuando se encuentran nuevos dispositivos de una investigación. Si una aplicación no tiene por qué esperar a que la investigación empiece, la API incluye el método

`retrieveDevices`(), que devuelve la lista de dispositivos que han sido encontrados en anteriores investigaciones o dispositivos que han sido clasificados como pre-conocidos. Los dispositivos pre-conocidos son aquellos que están definidos en el centro de control de Bluetooth como dispositivos con los que el dispositivo local contacta frecuentemente. Así, este método no realiza una investigación, pero proporciona una manera rápida de obtener una lista rápida de dispositivos que podrían estar en el área. Una vez que el dispositivo se descubre, se suele iniciar una búsqueda de servicios.

- Interfaz `javax.bluetooth.DiscoveryListener`

La interfaz `DiscoveryListener` tiene los siguientes métodos:

- `public void deviceDiscovered(RemoteDevice rd, DeviceClass c)`
- `public void inquiryCompleted(int c)`
- `public void servicesDiscovered(int transID, ServiceRecord[] sr)`
- `public void serviceSearchCompleted(int transID, int respCode)`

Los dos primeros métodos serán llamados durante el proceso de búsqueda de dispositivos. Los otros dos métodos serán llamados durante un proceso de búsqueda de servicios. El proceso de búsqueda de servicios se verá más adelante. Pasemos a investigar más profundamente los dos primeros métodos que son los usados durante una búsqueda de dispositivos.

- `public void deviceDiscovered(RemoteDevice rd, DeviceClass c)`

Cada vez que se descubre un dispositivo se llama a este método, que pasa dos argumentos. El primero es un objeto de la clase `RemoteDevice` que representa el dispositivo encontrado. El segundo argumento nos permitirá determinar el tipo de dispositivo encontrado

- `public void inquiryCompleted(int c)`

Este método es llamado cuando la búsqueda de dispositivos ha finalizado. Nos pasa un argumento entero indicando el motivo de la finalización. Este argumento podrá tomar los valores: `DiscoveryListener.INQUIRY_COMPLETED` si la búsqueda ha concluido con normalidad, `DiscoveryListener.INQUIRY_ERROR` si se ha producido un error en el proceso de búsqueda, o `DiscoveryListener.INQUIRY_TERMINATED` si la búsqueda fue cancelada.

Para realizar una búsqueda de servicios también usaremos la clase DiscoveryAgent e implementaremos la interfaz `DiscoveryListener`. En este caso nos interesarán los métodos `servicesDiscovered`() y `serviceSearchCompleted`() de la interfaz `DiscoveryListener`. Para comenzar la búsqueda usaremos `searchServices`() de la clase `DiscoveryAgent`.

- Clase `javax.bluetooth.DiscoveryAgent`

Las búsquedas de dispositivos y servicios Bluetooth las realizaremos a través del objeto `DiscoveryAgent`. Este objeto es único y lo obtendremos a través del método `getDiscoveryAgent`() del objeto `LocalDevice`:

```
DiscoveryAgent discoveryAgent = LocalDevice.getLocalDevice().getDiscoveryAgent();
```

A través de `DiscoveryAgent` tenemos la posibilidad de obtener un *array* de dispositivos que el usuario haya especificado como "ya conocidos" (*PREKNOWN*, quizá una lista de dispositivos "favoritos") o un *array* de dispositivos descubiertos en búsquedas anteriores (*CACHED*). Esto lo haremos llamando al método `retrieveDevices`() pasándole `DiscoveryAgent.PREKNOWN` o `DiscoveryAgent.CACHED` respectivamente. Este método no devolverá un *array* vacío si no encuentra dispositivos que coincidan con el criterio especificado, en vez de ello devolverá *null*.

El *array* devuelto por este método es de objetos `RemoteDevice`, los cuales, representan dispositivos remotos. La clase `RemoteDevice` permite obtener la dirección Bluetooth del dispositivo que representa a través del método `getBluetoothAddress`() en forma de *String*. También podremos obtener el "*friendly-name*" del dispositivo a través del método `getFriendlyName`(). Este

último método requiere un argumento de tipo booleano. Este argumento permite especificar si se debe forzar a que se contacte con el dispositivo para preguntar su "*friendly-name*" (true) o bien se puede obtener un "*friendly-name*" que tuvo en una ocasión previa (false). El método `getFriendlyName`() puede lanzar `una java.io.IOException` en caso de que no se pueda contactar con el dispositivo o este no provea un "*friendly-name*".
Antes hemos visto que el método `retrieveDevices`() no realiza una búsqueda, sino que nos devuelve un *array* de dispositivos remotos ya conocidos o encontrados en búsquedas anteriores. Para comenzar una nueva búsqueda de dispositivos llamaremos al método `startInquiry`(). Este método requiere dos argumentos. El primer argumento es un entero que especifica el modo de conectividad que deben tener los dispositivos a buscar. Este valor deberá ser `DiscoveryAgent.GIAC` o bien `DiscoveryAgent.LIAC`. El segundo argumento es un objeto que implemente `DiscoveryListener`. A través de este último objeto serán notificados los dispositivos que se vayan descubriendo. Para cancelar la búsqueda usaremos el método `cancelInquiry`().

- Clase `javax.bluetooth.DataElement`

Esta clase puede tener varios tipos de datos que un atributo puede tomar. Puede ser:

- Enteros con o sin signo que tienen uno, dos, cuatro, ocho o dieciséis bytes de longitud.
- *String*
- *Boolean*
- UUID
- Secuencias de cualquiera de alguno de esos tipos escalares

La clase también representa una interfaz para construir y recuperar el valor de un atributo de servicio.

Capítulo 5. XACML

XACML [17] es un estándar cuya responsabilidad recae en el consorcio OASIS (*Organization for the Advancement of Structured Information Standards*), que describe tanto un lenguaje de políticas como un protocolo petición/respuesta para el control de las decisiones de autorización. Ambos, lenguaje y protocolo, están definidos en XML. El lenguaje de políticas XACML se utiliza para describir requisitos de control de acceso generales, y tiene unos puntos de extensión estándar que permiten la definición de nuevas funciones, tipos de datos, combinaciones lógicas, etc. El protocolo petición/respuesta nos permite componer una petición para solicitar si cierta acción debería, o no, ser permitida, además de ofrecernos formas de interpretar las decisiones de control. La respuesta siempre incluye una contestación que refleja si la petición debería ser o no permitida utilizando uno de los siguientes cuatro valores: *Permit, Deny, Indeterminate* (no se pudo tomar una decisión porque ocurrió un error o fue omitido algún valor requerido) o *NotApplicable* (la petición no puede ser contestada por este servicio). Estos va lores se encuentran predefinidos por la propia especificación.

La situación típica es que alguien quiera realizar cierto conjunto de acciones sobre un recurso. Realizará una petición al servicio que realmente protege el recurso (como un sistema de ficheros o un servidor Web), que se denomina Punto de Aplicación de la Política o PEP (*Policy Enforcement Point*). El PEP formará una petición basada en los atributos del solicitante, el recurso en cuestión, la acción y otra información que pertenezca a la petición. El PEP enviará la petición al PDP que deberá ser quién evalúe si la petición debe ser aceptada o denegada, como podemos observar en la figura 17.

Existen muchos lenguajes específicos de aplicación o propietarios que ya realizan las mismas acciones, aunque XACML posee varios puntos a su favor:

- Es un estándar. Utilizar un lenguaje estándar supone utilizar algo que ya ha sido revisado por una comunidad de expertos y usuarios muy grande, evitándonos tener que pensar sobre los todos los puntos, incluidos los más críticos, implicados en diseñar nuestro propio lenguaje. Además, como XACML se está utilizando cada vez más, será más sencillo interactuar con otras aplicaciones utilizando el mismo lenguaje.

- Es genérico. Esto significa que en vez de intentar proporcionar control de acceso para un entorno en particular o un tipo de recurso específico, puede ser utilizado

dentro de cualquier entorno. Una política puede ser escrita de forma que puede ser utilizada por muchas aplicaciones diferentes, y como se hace uso de un lenguaje común, la gestión de la política se simplifica enormemente.

- Es distribuido. Esto significa que una política puede ser escrita de forma que puede referenciar otras políticas ubicadas en localizaciones arbitrarias. El resultado es que en vez de tener que gestionar una única política monolítica, diferentes agentes o grupos pueden gestionar sub-piezas de las políticas de manera apropiada, y XACML conoce cómo combinar correctamente los resultados de estas diferentes políticas para tomar una única decisión.

- Es poderoso. Pese a que existen múltiples maneras de extender el lenguaje base, muchos entornos no necesitarán hacerlo. El lenguaje estándar ya soporta una amplia variedad de tipos de datos, funciones, y reglas para combinar los resultados de las distintas políticas. Además, ya existen muchos grupos estándares que se encuentran trabajando sobre extensiones y perfiles que permitirán "enganchar" XACML con otros estándares como SAML (*Security Assertion Markup Language*) o LDAP (*Lightweight Directory Access Protocol*), cosa que incrementará el número de formas en el que XACML puede ser utilizado.

Figura 17: Contexto XACML [18]

5.1 Policy y PolicySet

La raíz de todas las políticas XACML es un elemento *Policy* o *PolicySet*. Un elemento *PolicySet* es un contenedor que puede contener otros elementos *Policy* o *PolicySet*, así como referencias a políticas que se encuentran en ubicaciones remotas. Un elemento *Policy* representa una única política de control de acceso, expresada a través de un

conjunto de reglas Rules. Cada documento de política XACML contiene exactamente un elemento raíz *Policy* o un *PolicySet*.

Puesto que un elemento *Policy* o *PolicySet* puede contener múltiples *Rules*, cada una de ellas podrían evaluarse a diferentes decisiones de control de acceso. XACML necesita alguna forma de reconciliar las decisiones que produce cada regla. Esto se realiza mediante una colección de elementos que permiten reflejar una combinación de algoritmos (*Combining Algorithms)*. Cada algoritmo representa una forma distinta de combinar las múltiples decisiones para alcanzar una decisión única. Existen Algoritmos de Combinación de Políticas (utilizados por los elementos *PolicySet*) y Algoritmos de Combinación de Reglas (utilizado por una política *Policy*). Estos Algoritmos de Combinación se utilizan para construir incrementalmente políticas complejas y, mientras existen un total de siete algoritmos estándar, esto no impedirá que podamos construir nuestro propio "juego" de algoritmos adecuados a nuestras necesidades.

5.2 Objetivos y reglas

Parte de lo que un servicio PDP (*Policy Decision Point*) de XACML necesita hacer es buscar una política que sea aplicable a una petición dada. Para conseguir esto, XACML proporciona otra característica denominada Target. Un Target es básicamente un conjunto de condiciones simplificadas para el sujeto (elemento *Subject*), recurso (elemento *Resource*) o la acción (elemento *Action*) que deben ser expresadas por un *PolicySet*, *Policy* o Rule de forma que puedan ser aplicadas sobre una petición dada.
Es decir, para poder determinar si cierta política o conjunto de políticas, o si cierta regla aplica a la petición XACML recibida se deberá analizar el contenido del elemento Target que contiene el sujeto, el recurso y la acción sobre la que aplican.

Las condiciones utilizan funciones "booleanas" para comparar los valores encontrados en una petición con aquellos incluidos en un Target. Si todas las condiciones del Target se ven cumplidas, entonces su elemento *PolicySet*, *Policy* o *Rule* asociadas es aplicable a la petición. Además de ser una manera de comprobar la aplicabilidad, la información del Target también proporciona una forma de "indexar" las políticas, lo que resulta muy útil si poseemos muchas políticas y necesitamos realizar una búsqueda rápida a través de todas ellas para ver cuál es la que aplica.

Una vez se encuentra una política que aplica a la petición, se procede a evaluar las reglas *Rule* que contiene. Como ya hemos dicho, XACML se basa en políticas que

contienen reglas. El resultado de combinar los resultados de evaluar cada una de las reglas contenidas en la política será el que refleje si, para esa política, la petición puede ser autorizada o no.

Una política puede tener cualquier número de reglas *Rule* las cuales contienen la lógica principal de una política XACML. El corazón de la mayor parte de las reglas es un elemento *Condition*, que es una función "booleana". Si la condición reflejada en el elemento *Condition* se evalúa como verdadero, entonces es devuelto el valor *Effect* de la regla (un valor '*Permit'* o '*Deny'* que está asociado con la evaluación con éxito de la regla). La evaluación de una condición también puede resultar en error (devolviendo entonces el valor '*Indeterminate'*) o en el descubrimiento de que esa *Condition* no aplica a la petición (devolviendo en este caso el valor '*NotApplicable'*). Una condición *Condition* puede ser bastante compleja y puede estar construida a partir de un conjunto arbitrario de atributos y funciones "booleanas". En resumen, el corazón las políticas XACML son las reglas que las componen, y el corazón de las reglas las condiciones que contienen.

5.3 Atributos, Valores de Atributos y Funciones

La divisa con la que XACML trata son los atributos. Los atributos son valores con nombre de tipos conocidos que podrían ser cosas como un identificador de un emisor o una fecha y hora de emisión. Específicamente, los atributos son características del sujeto, del recurso, de la acción o del entorno para el cual la petición de acceso ha sido realizada. Un nombre de usuario, el fichero al que se quiere acceder y la hora del día son todos posibles valores de atributos. Cuando se envía una petición XACML desde un PEP a un PDP, ésta se compondrá básicamente de atributos que serán comparados con los valores de los atributos en una política, pudiéndose llevar a cabo la toma de decisión de acceso.

Una política resuelve los valores de los atributos procedentes de una petición, o de alguna otra fuente (Punto de Recuperación de Información), mediante dos mecanismos: el elemento *AttributeDesignator* y el elemento *AttributeSelector*. El elemento *AttributeDesignator* permite a la política especificar un atributo dado un nombre, un tipo y, opcionalmente, un emisor del atributo. El PDP entonces buscará el valor de dicho atributo en la petición, o en cualquier otro lugar en el caso de que no se encontraran valores coincidentes en la petición. Existen cuatro tipos de *Designators*, uno para cada uno de los posibles tipos de atributos de una petición: *Subject*, *Resource*, *Action* y *Environment*. Puesto que los atributos *Subject* pueden descomponerse en sub-

categorías, se pueden especificar también elementos *SubjectAttributeDesignators* como una categoría en la que buscar los valores.

Por su parte, el elemento *AttributeSelectors* permite a una política buscar los valores de los atributos mediante una petición *XPath*. Ambos elementos, *AttributeDesignator* y *AttributeSelector* pueden retornar múltiples valores (ya que podrían coincidir varios atributos en una petición) y, por lo tanto, XACML proporciona un tipo de atributo especial denominado *Bag* (Bolsa). Los elementos *Bags* son una colección desordenada que permiten duplicados, y son siempre los que los selectores devuelven, incluso si solamente coincidiera un valor. En el caso en que no se dieran coincidencias, se devolverá un elemento *Bag* vacío.

Bloque III

Capítulo 6. Descripción general del sistema

Para tener una visión general y entender mejor la descripción del sistema, debemos observar la figura 18. En ella, se distinguen los principales bloques de los que consta el proyecto.

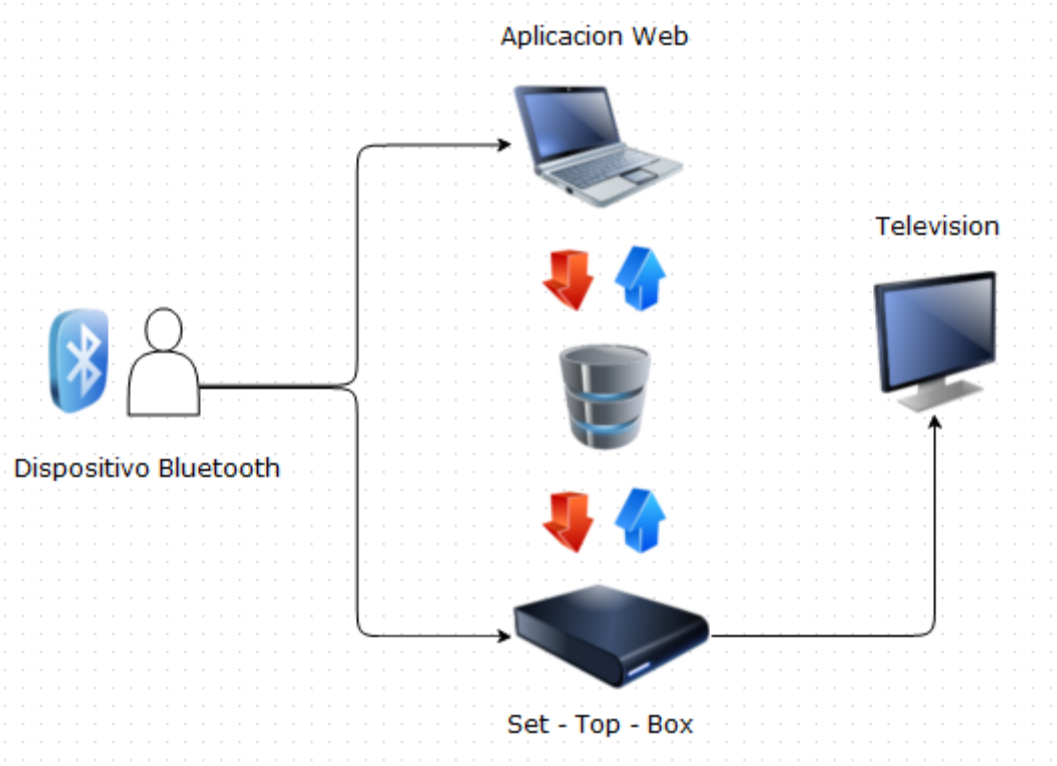

Figura 18: Diagrama del sistema

El primer bloque se corresponde con el usuario. Puede tratarse de cualquier miembro de la familia que posea un dispositivo Bluetooth, este activo y/o registrado, y se encuentre dentro del rango tanto de la aplicación web como del *set-top-box*.

El segundo bloque correspondería con la aplicación web. A ella pueden acceder tanto los usuarios nuevos que se dispongan a crearse un perfil de acuerdo a sus preferencias y gustos, como los usuarios ya registrados que quieres modificar alguna de sus elecciones, o darse de baja en el servicio.

El tercer bloque quedaría reservado para el *set-top-box*, el cual, al igual que la aplicación web, interactúa con los ficheros de almacenamiento de datos. Acto seguido, adaptaría el contenido de la televisión a los usuarios registrado y activos en ese momento. En nuestro caso, haremos uso del emulador XletView que nos permite, además de ejecutar aplicaciones en código Java, visualizar resultados por pantalla, como lo haría un dispositivo en el salón de una casa.

6.1 Diseño del sistema

En la figura 19 podemos observar la arquitectura software de este sistema de personalización de guías de televisión digital terrestre, que se compone de una serie de bloques de software interconectados de manera que distribuyen toda la funcionalidad en forma de módulos, de acuerdo al artículo [19]. Si bien es cierto que se intentó seguir todas las pautas en cuanto al diseño, se tuvieron que tomar ciertas decisiones para resolver problemas y cuestiones que estaban abiertas o no contempladas en un primer momento.

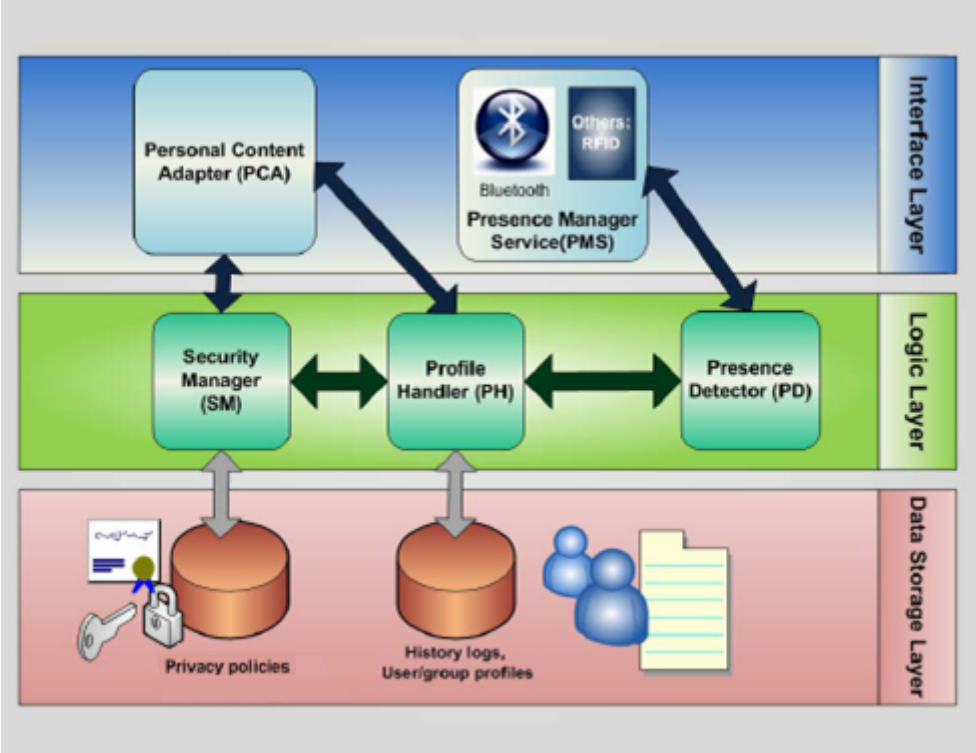

Figura 19: Arquitectura en capas [19]

6.2 Capa de presentación

La capa de presentación se corresponde con la parte superior de la arquitectura y consiste en el manejo de interfaces de usuario compuesta por el adaptador de contenido personal (PCA) y el administrador de servicios de presencia (PMS).

Por un lado, el modulo PCA ofrece una interfaz web fácil de usar para la configuración del sistema, entendiendo por configuración funciones como el registro, la modificación y la eliminación de usuarios, siempre asociados a un identificador Bluetooth, así como la puesta en marcha de las políticas de seguridad. A estas tareas se puede acceder a través de la televisión por medio del explorador.

Por otro lado el modulo PMS proporciona un servicio administrador de presencia que gestiona los criterios para detectar que usuarios se encuentran en la sala. La tecnología usada, por ser la más extendida en la actualidad ha sido Bluetooth, aunque bien podría a haber sido también la identificación por radio frecuencia (RFID).

6.2.1 Adaptador de Contenido Personal (PCA, *Personal Content Adapter*)

Este módulo se encarga de la adaptación del contenido a los gustos del espectador, lo que implica una adaptación a las preferencias y a las políticas de seguridad del usuario. Esta adaptación puede ser explícita o implícita. En el primer caso, el sistema recibe información a través de las preferencias del usuario. Sin embargo, estos métodos no pueden adaptar el contenido dinámicamente hasta que los cambios no son detectados, a menos que el usuario los provea de forma explícita.

En lo que respecta al funcionamiento, el PCA recoge la información de la emisión, aplica una serie de filtros según las características del perfil del usuario y, tras las políticas de seguridad obtenidas del *Profile Handler*, muestra el contenido que mejor se ajuste al usuario.

6.2.2 Administrador de Servicios de Presencia (PMS, *Presence Manager Service*)

El administrador de servicios de presencia ofrece una interfaz de detección de usuarios y notificación de eventos, con el objetivo de mantener el contexto de esa presencia en todo momento. Actualmente existen muchas técnicas de detección que nos permiten saber quien se encuentra frente al televisor. Entre ellas, destacan:

- Bluetooth: protocolo para el intercambio de datos a corta distancia para dispositivos móviles y fijos. Las soluciones que nos aporta el uso de esta tecnología son de costes bajos y de un bajo consumo de recursos, aunque por el contrario, podemos encontrar limitaciones en torno a las distancias de uso, quedando limitada la tecnología a 10 metros como máximo, con un funcionamiento más que efectivo a los 2 y 3 metros.

- RFID: tecnología que utiliza la comunicación por radiofrecuencia a través del intercambio de datos entre un lector y una etiqueta electrónica vinculada a un objeto, con el fin de conseguir un seguimiento e identificación del mismo. La principal ventaja que ofrece esta tecnología es que tanto el sistema como los dispositivos usados en él son de bajo coste, ligeros y pequeños. Por el contrario, ofrece distancias muy en cuanto a las distancias máximas, que rondan los 1 y 2 metros.

- Interacción Natural: hace referencia al concepto de interacción entre los sentidos de un ser humano y un dispositivo, ya sea mediante voz, reconocimiento facial u otros, que permite la identificación de usuarios. La principal ventaja de los dispositivos de interacción humana es que los usuarios no están obligados a llevar ningún dispositivo personal encima, pero no ofrece una comunicación a distancia.

Una de las ventajas de RFID frente a Bluetooth es que no se necesita pasar por el trámite y pasos necesarios para emparejar dispositivos, así como el bajo consumo de energía que requiere RFID con respecto a Bluetooth. Por otro lado, la actual tecnología desarrollada para la interacción natural ofrece atractivas características para la detección de presencia y la identificación con una alta precisión.

Para nuestro propósito, vamos a hacer uso de Bluetooth, por varias razones. Se trata de una tecnología de fácil desarrollo, ampliamente adoptada y nos proporciona la distancia necesaria para que la comunicación entre el dispositivo y el sistema se lleve a cabo (desde el sofá al televisor). Además, hará falta que cada usuario del sistema este en posesión de un aparato móvil que disponga de Bluetooth, algo prácticamente implementado en cualquier móvil hoy en día.

Pero nuestro enfoque no es perfecto, ya que para que la detección se produzca, el dispositivo tiene que estar en el radio de acción del sistema, y esto implica que pueda reconocer dispositivos que se encuentren en habitaciones cercanas. Es por ello que un objetivo futuro será mejorar el sistema, con algún otro método de detección o la incorporación de accesorios que se adapte a todas y cada una de las circunstancias que se puedan dar, como por ejemplo la colocación de una cámara.

6.3 Capa de lógica de negocio

Formada por tres componentes: el detector de presencia (PM), el controlador de perfiles (PH) y el gestor de seguridad (SM).

6.3.1 Detector de Presencia (PD, *Presence Detector*)

Consiste en aplicaciones cliente entre el dispositivo y el sistema (*set-top-box*). La primera vez que un usuario se registra en el sistema, lo hace asociando su perfil al dispositivo que va a ser usado para la detección. El objetivo de este módulo es la obtención en todo momento de las direcciones MAC de los dispositivos que se encuentran en el radio de acción de la aplicación Bluetooth localizada en el *set-top-box*.

Además, el detector de presencia está asociado al controlador de perfiles, para llevar a cabo la gestión de estos. Como la detección de usuarios es automática, si detecta más miembros de la familia delante del televisor, se lleva a cabo una adaptación automática del contenido. La idea es que el sistema sea capaz de reconocer automáticamente a un usuario de la familia cuando se aproxime. De igual manera, cuando un usuario se aleja del sistema, el detector de presencia detecta el evento y reconfigura automáticamente su apariencia.

6.3.2 Controlador de perfiles (PH, *Profile Handler*)

El controlador de perfiles construye una tabla de asociación de manera que cada dirección MAC Bluetooth se asigna a un id, correspondiente al nombre o alias de ese dispositivo Bluetooth. Cuando el controlador de perfiles recibe una notificación del detector de presencia, se va a la tabla de asociación para recuperar el perfil del usuario y acto seguido aplicar la política de seguridad correspondiente.

Por otra parte, el sistema también contempla la posibilidad de que varios usuarios coincidan delante del televisor, situación que se resuelve con un sistema de filtrado basado en una adaptación de los perfiles a unas características comunes. De esta manera, el controlador de perfiles es capaz de construir una serie de filtros capaces de adaptar el contenido total de canales y programas al perfil o perfiles detectados.

La cantidad de usuarios está limitada por la cantidad de miembros de la familia, y cada perfil está definido por unas características como la edad, canales favoritos, preferencias de género, etc. Dividiremos esta información de perfil en dos subconjuntos: un primero formado por datos relacionados con la seguridad, y un segundo formado por datos para la configuración del contenido.

El primero de los subconjuntos se lleva a cabo a partir de la edad del usuario y de la hora de entrada en el sistema. Si se trata de un usuario que no supera los 18 años y nos encontramos en una franja horaria de entre las 23:00 de la noche y las 7:00 de la mañana, la política de seguridad no nos permitirá hacer uso del sistema.

En cuanto al segundo, son datos que se han recopilado a través de la interfaz web, y tras la aplicación de un filtrado que actúa de la siguiente manera. En un primer momento, se lleva a cabo un filtrado de la EPG por la edad, manteniendo todos los programas de todos los canales en caso de tratarse de un usuario mayor de edad, o por el contrario eliminando los programas estén recomendados para usuarios con más de 18 años. Acto seguido, sobre los programas restantes se aplican las preferencias de canal, descartando así programas, para, sobre la lista reducida, aplicar el filtro de género. Solo nos faltaría quedarnos con los programas de la lista reducida que estén en ese momento en emisión. De esta manera se crea un Vector ordenado de programas según nuestras preferencias, que nos servirá para acto seguido configurar toda la interfaz grafica de la EPG a mostrar.

El filtrado de género se realiza según un orden, establecido en el momento de registrar un usuario en el sistema. El género con mayor prioridad, obtendrá un trato de favor frente a los restantes, de manera que la EPG mostrara un contenido escalonado, con los programas que más se adapten a nuestros gustos en primer lugar. Por el contrario, los programas filtrados con géneros a los que les hemos otorgado menos prioridad aparecerán en las posiciones más bajas.

El registro de grupos de usuario no hace falta llevarlo a cabo, ya que ante la situación de tener a más de una persona delante del televisor, comenzaría una comparación de las preferencias de ambos perfiles para ver cuáles de ellas son comunes y así poder mostrar una EPG que satisfaga a ambos. En el caso de encontrarse un menor delante del sistema con mas usuario siendo estos últimos mayores de edad, se aplicaría la política de seguridad para el menor de 18 años, por lo que si el intervalo fuera el de prohibición, el sistema no funcionaria. Si por el contrario estamos en el horario adecuado para el menor, solo se mostrara el contenido de este.

En la figura 20 podemos observar el flujo a seguir para mostrar la guía electrónica de programas personalizada.

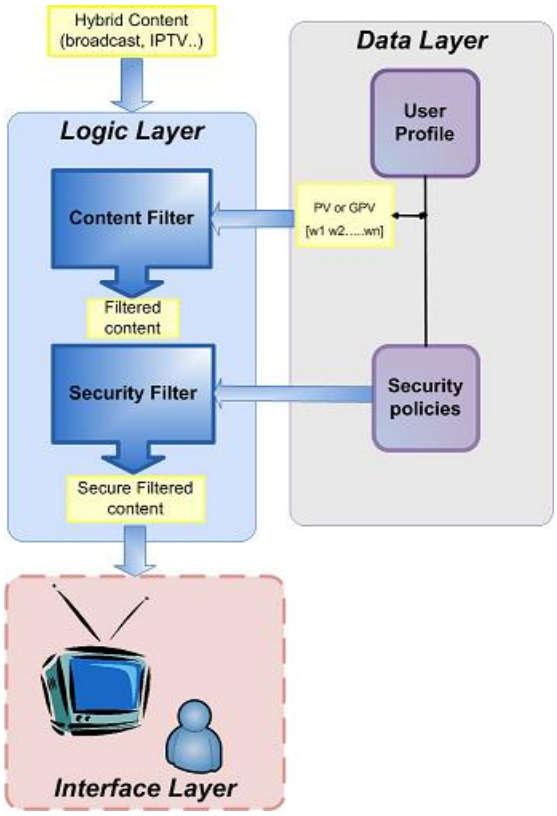

Figura 20: Proceso de filtrado para mostrar una EPG personalizada [19]

6.3.3 Gestor de Seguridad (SM, *Security Manager*)

Modulo que se encarga de gestionar los distintos aspectos de la seguridad, está relacionado con el controlador de perfiles con el fin de determinar la seguridad más adecuada de configuración para el usuario actúalo grupo de usuarios. Entre las funciones principales de este modulo, encontramos:

- Políticas de seguridad: creadas y administradas por este módulo, garantizan que se encuentra activo un contexto de seguridad adecuado, como por ejemplo prevenir que los niños vean contenidos que no son apropiados por la hora a la que están viendo la televisión.

- Gestión de sesiones: con el fin de crear y mantener un contexto de seguridad en cada momento, el gestor de seguridad también administra las sesiones, de manera que pueden cambiar de forma dinámica y readaptarse en función de los usuarios actuales.

6.4 Capa de persistencia

La capa de persistencia se encarga del almacenamiento de los datos para que el sistema funcione, esto es: usuarios y perfiles, políticas de seguridad, *logs* en el sistema, etc. Los usuarios deben registrar sus usuarios antes de comenzar a usar el sistema a través de una interfaz web. En primer lugar se le piden datos personales como la edad y el nombre, además de una serie de preferencias, para elaborar el filtro de seguridad. Como hemos indicado anteriormente, no es necesario establecer preferencias en grupos, dado que el sistema automáticamente realiza unas comparativas para determinar unas preferencias acordes a todos los gustos.

Con el fin de llevar un registro de las entradas en el sistema de los distintos usuarios, se almacenan los *logs* de los distintos perfiles a lo largo del tiempo. En caso de que fuese un grupo el que se dispusiese a ver la tele, en el log quedarían reflejados los identificadores de los distintos dispositivos que estuviesen al mismo tiempo delante del televisor, compartiendo hora de entrada en el sistema.

Por último, las políticas de seguridad son una serie de reglas a configurar que evalúan condiciones basadas en los atributos o características de los distintos perfiles, con el fin de permitir o denegar el acceso al sistema de los distintos usuarios. Además, para preservar la seguridad, el almacenamiento de todos los datos e lleva a cabo en el sistema local, para así preservar la privacidad de los usuarios.

Capítulo 7. Implementación del sistema

Una vez definido el diseño del sistema, procederemos a su implementación. Se trata de implementar cada uno de los módulos de tal manera que satisfagan sus especificaciones y funcionalidades, utilizando la tecnología adecuada.

Vamos a describir a continuación, de manera más detallada cada uno de los módulos con un punto de vista centrado más en el desarrollo y en la programación de los mismos.

7.1 Esquema de la implementación

Disponemos de tres capas en las que tenemos estructurado el proyecto, cada una de ellas con sus correspondientes módulos y la forma en la que se ha desarrollado:

- **Capa de presentación**

    - Módulo Adaptador de Contenido Personal: se trata del servicio web, encargado de recopilar datos como la edad, el nombre, y las distintas preferencias de los diferentes usuarios que van a hacer uso del sistema. Para ello, hemos hecho uso de Eclipse y Java EE. El servicio corre sobre un servidor Apache, donde el intercambio de información se lleva a cabo entre el servidor (*servlets*) y el cliente (JSP y *JavaScript)*. Una vez recopilados los datos necesarios acerca de los gustos del usuario, se comunica con el modulo controlado de perfiles, para que cree una nueva entrada en la tabla de asociación. Además, obtiene información de éstos a través del controlador de perfiles para la modificación de datos, por ejemplo. La elección de un servicio web se debe a su facilidad en el uso y en la presentación de datos y recursos al usuario. Además, el poder desplegar el servidor sobre el que corre la aplicación en nuestro local, nos va a proporcionar un extra en cuanto a la privacidad de los usuarios, evitando posibles ataques que se podrían producir si la aplicación estuviese desplegada en un servidor compartido con otros sistemas.

- Módulo Administrador de Servicios de Presencia: se trata de una interfaz que varía en función de si algún espectador o espectadores se encuentran en el radio de acción de la tecnología Bluetooth ubicada en el *set-top-box* y para ello hemos hecho uso de un emulador de *set-top-box*, en concreto de XletView, para poder ejecutar la aplicación que gestiona el sistema (Xlet). Si algún usuario es detectado, la pantalla que nos proporciona este programa cambia su contenido, adaptándolo a partir de los datos que el controlador de perfiles ha obtenido de la dirección MAC proporcionada por el modulo detector de presencia, el cual se comunica directamente con PMS. Además, la pantalla nos ofrece cierta interacción a través de los botones, lo que implica la comunicación con la lógica de negocio en este sentido.

- **Capa de lógica de negocio**

  - Módulo Detector de Presencia: dentro de la lógica de negocio, se trata de la parte del sistema que implementa la tecnología Bluetooth, a través del paquete `javax.bluetooth` que nos proporciona JAVA. Entre las distintas clases que nos proporciona, encontramos `DiscoveryListener`, que nos proporciona los métodos necesarios para la detección de dispositivos, como `deviceDiscovered` e `inquiryCompleted`, entre otros. Una vez obtenidos el identificador y la dirección MAC del dispositivo, envía esta información al módulo controlador de perfiles para que obtenga las preferencias del usuario. Además, este modulo se encarga de comprobar si el dispositivo que ha activado el sistema se encuentra siempre activo, manteniendo su sesión.

  - Módulo Controlador de Perfiles: también dentro de la lógica de negocio, se encarga de gestionar la tabla de asociaciones, realizando consultas y creando o borrando entradas. Se trata de código Java para la lectura y tratamiento de archivos XML, y para ello optamos por las clases que nos proporciona el API JDOM (*Java Document Object Model*). Las peticiones de consultas, modificaciones o eliminaciones de la tabla de asociaciones provienen del modulo PCA, mientras que del modulo PD solo entran peticiones de consulta, para la modificación del contenido de la pantalla del Xlet de acuerdo a la dirección MAC del dispositivo.

  - Módulo Gestor de Seguridad: encargado de la seguridad del sistema, contiene la lógica de negocio que vela por el correcto uso del sistema de acuerdo a unos parámetros establecidos. Realiza una petición XACML, esperando una respuesta de acceso o denegación, según datos usados en la petición y recogidos del

modulo controlador de perfiles. Acto seguido, detiene o continua con la ejecución de la aplicación según los resultados.

- **Capa de persistencia**

Compuesta por ficheros que hacen la funcionalidad de bases de datos, podemos dividirla en tres:

- Histórico de *logs*: archivo donde quedan registrados todos los usuarios que han hecho uso del sistema, así como el tiempo que estuvieron usándolo.
- Ficheros que contienen la política de seguridad: se trata de ficheros del estándar XACML que definen la sintaxis y la semántica de un lenguaje basado en XML para especificar y evaluar las políticas de acceso, en este caso, a nuestro sistema.
- Fichero de perfiles: archivo que contiene los distintos perfiles registrados por el sistema, asociados cada uno de ellos con un identificador Bluetooth y unas preferencias acordes a los gustos de los usuarios.

7.2 Aplicación Web

La aplicación web va a estar compuesta por una serie de *servlets*, JSP y código *JavaScript* [20], que nos aportaran la funcionalidad requerida para el registro, modificación y eliminación de perfiles.

Los *servlets* se encargan de recibir los datos de los formularios para procesarlos y realizar las acciones correspondientes. Disponemos de 4 *servlets*, tres de ellos se encargan de cargar datos que van a ser necesarios en sus respectivas pantallas, como son los identificadores de los dispositivos Bluetooth activos en el momento para el caso del registro de usuarios, o los identificadores de los dispositivos ya registrados para las pantallas de modificación y eliminación (tabla de asociación). El cuarto *servlet* se encarga de una vez rellenados los formularios en el registro y la modificación, realizar las modificaciones pertinentes en las tablas de asociación.

En la figura 21 se puede observar el diagrama de flujo de la aplicación web, donde se muestra el funcionamiento más detallado.

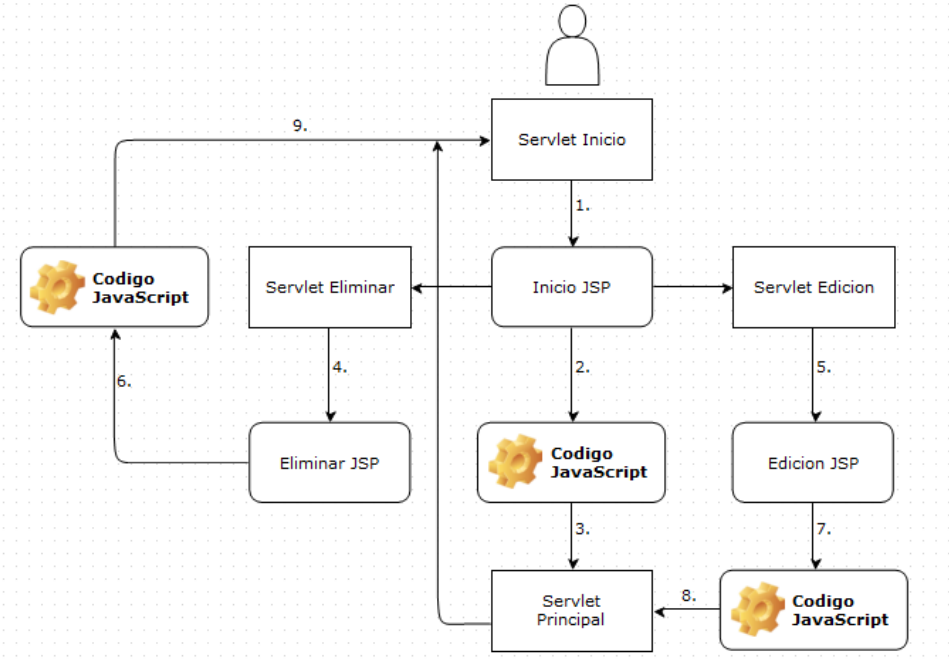

Figura 21: Diagrama de flujo de la aplicación Web

Los pasos que se definen en la figura corresponden a:

1. Listado de los dispositivos Bluetooth activos en el momento y los parámetros de las preferencias.

2-6-7. Recogida de los datos del formulario.

3-8-9. Validación de los datos del formulario.

4. Listado de los dispositivos Bluetooth existentes en la tabla de asociación.

5. Listado de los dispositivos Bluetooth existentes en la tabla de asociación y parámetros de las preferencias.

Como tabla de asociación, se utiliza un archivo XML, al que hemos denominado perfiles.xml (véase figura 22). Cada vez que registramos un nuevo usuario, se crea una entrada con todos los parámetros recogidos, entre los que podemos distinguir:

- La raíz, denominada perfiles, con distintos elementos de tipo perfil.
- Cada uno de los perfiles registrados cuenta con atributos como son la edad, el nombre o alias, y la MAC del dispositivo, y una serie de elementos entre

los que se encuentran los géneros elegidos con el atributo prioridad asignada y los canales seleccionados como preferidos por el usuario.

```
<?xml version="1.0" encoding="UTF-8"?>
<perfiles>
  <perfil edad="24" nombre="Javier" mac="002608C09464">
    <generos>
      <genero prioridad="0">Documental</genero>
      <genero prioridad="1">Informativos</genero>
      <genero prioridad="2">Cine</genero>
      <genero prioridad="3">Dibujos</genero>
      <genero prioridad="4">Entretenimiento</genero>
      <genero prioridad="5">Deportes</genero>
    </generos>
    <canales>
      <canal>NatGeographic</canal>
      <canal>VayaCine</canal>
      <canal>Nova</canal>
      <canal>Neox</canal>
      <canal>FDF</canal>
      <canal>TeleDeporte</canal>
      <canal>EuroSport</canal>
    </canales>
  </perfil>
</perfiles>
```

Figura 22: Fichero de perfiles

La lectura de los distintos canales de la EPG por defecto se realiza a partir de un archivo XML, como el de la figura 23, donde disponemos de manera actualizada la programación correspondiente a cada semana de todos los canales de la televisión digital terrestre de nuestro país. En la figura podemos observar cómo está estructurada esta EPG.

```
<tv>

    <programa  tipo ="Documental" inicio="09:00" stop="10:30" channel="Discovery">
    <title>Top Gear</title>
    <descripcion>coches</descripcion>
    <credits>
    <actor>Maria Sarmiento</actor>
    </credits>
    <date>2012</date>
    <category>Documental</category>
    <capitulo-num system="xmltv_ns">6</capitulo-num>
    </programa>

    <programa tipo ="Documental" inicio="10:30" stop="11:30" channel="Discovery">
    <title>Bear Gryll</title>
    <descripcion>Aventuras</descripcion>
    <credits>
    <actor>Ana Blanco</actor>
    </credits>
    <date>2012</date>
    <category>Documental</category>
    <capitulo-num system="xmltv_ns">393</capitulo-num>
    </programa>
```

Figura 23: Fichero que contiene la EPG

7.2.1 Interfaces

Para el registro, modificación o eliminación de los usuarios se utilizan tres páginas JSP, cada una con una funcionalidad distinta, y los *servlets* que se encuentran tras ellas. Además, disponemos de código *JavaScript* dentro de los JSP, para capturar eventos como cliqueo de botones, validación de campos o generación de errores a la hora de introducir mal los datos, por ejemplo.

Como podemos observar en la figura 24, la pantalla correspondiente al registro de miembros nos ofrece diversos campos a rellenar para completarlo:

- Nombre: donde podemos seleccionar de entre los identificadores de los dispositivos que el sistema ha detectado activos en el momento.
- Edad
- Géneros: con sus correspondientes prioridades para formar el vector.
- Canales favoritos: de entre los canales que nos ofrece la guía de programas por defecto, podemos seleccionar cuales son nuestros preferidos por los programas que ofrecen.

Figura 24: Pantalla registro de usuarios

Además, la pantalla dispone de un combo que refleja cómo se va rellenando el vector de prioridades de los géneros. Dispone también de un botón que permite limpiarlo, en caso de que hubiéramos errado en la elección y quisiéramos rectificar.

La figura 25 corresponde a la pantalla de edición de miembros, donde un desplegable nos listará los identificadores de los dispositivos ya registrados en el sistema, para seleccionar el nuestro y comenzar así la modificación de las preferencias.

Figura 25: Pantalla edición de usuarios

En la figura 26 podemos observar como es la pantalla de eliminación de miembros, semejante a la anterior, donde un desplegable nos lista los identificadores de los dispositivos ya registrados para acto seguido proceder a su eliminación.

Figura 26: Pantalla eliminación de usuarios

En caso de producirse algún tipo de error al rellenar los campos del formulario de registro o bien que hayamos obviado alguno de ellos, saltará un error como el de la figura 27.

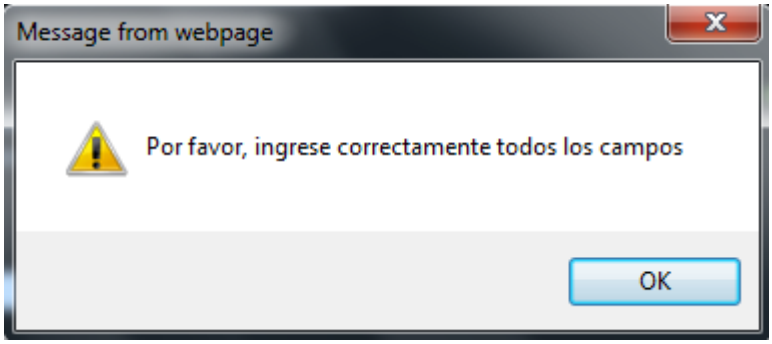

Figura 27: Error en la validación de los datos

7.3 XletView

Para la ejecución de la aplicación que va a gestionar y soportar parte de nuestro sistema, se ha utilizado un emulador de Xlets, que nos va a permitir implementar toda nuestra lógica de negocio, además de proporcionarnos una interfaz visual que hará las funciones de pantalla de televisión.

7.3.1 Módulos

- Módulo Administrador de Servicios de Persistencia

Contiene la clase principal *XletviewApplication* (figura 28), que gestiona el ciclo de vida de nuestra aplicación, y genera el contenido que mostraremos por pantalla. Además, nos permite capturar eventos como el pulsado de teclas, muy útil a la hora de avanzar de página o de retroceder en nuestra guía de programas personalizada.

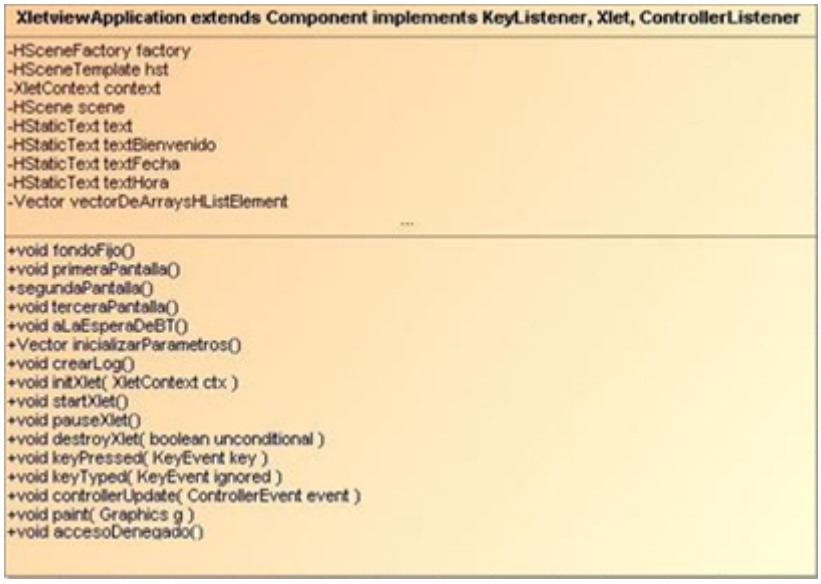

Figura 28: Clase XletViewApplication

- Modulo Detector de Presencia

Entre otras, contiene la clase *RemoteDeviceDiscovery* , la cual devolverá una lista de los nombres de los distintos Bluetooth que se encuentran activos con su MAC correspondiente, y la clase *Sesion*, que realiza comprobaciones para determinar si el usuario sigue estando delante del televisor. En la figura 29 podemos observar el diagrama de clases de este módulo.

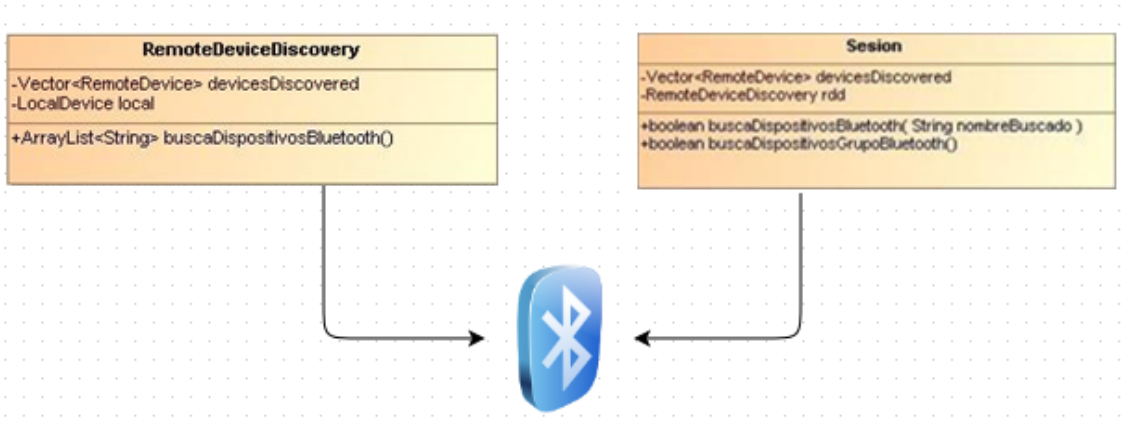

Figura 29: Módulo detector de presencia

- Modulo Controlador de Perfiles

Contiene la clase *Perfiles*, que determina qué usuarios del sistema están activos y registrados (operaciones de lectura), y realizará un filtrado a partir de sus preferencias, confeccionando una lista ordenada con su EPG personalizada, así como las clases *LectorTodosCanales*, *LectorPerfiles*, *LectorGenerosPerfil*, *LectorCanalesPerfil* y *LectorCanales* (figura 30), que dadas una serie de parámetros establecidos en el constructor o pasados en métodos como pueden ser una edad, un nombre y/o una MAC, devuelven una serie de vectores o listas específicos.

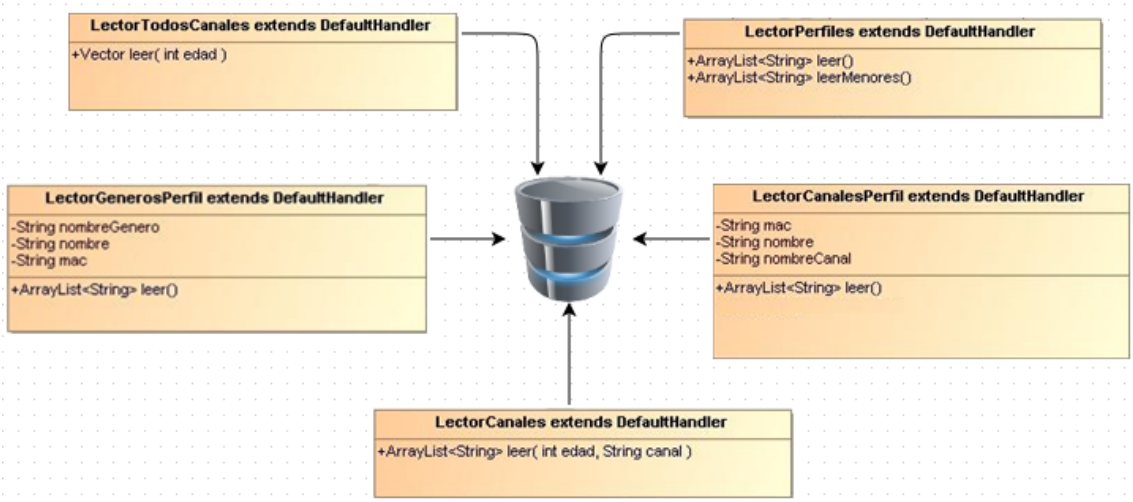

Figura 30: Módulo controlador de perfiles

- Módulo Security Manager

Contiene la clase *PoliticaAcceso* (figura 31), la cual determina si para un usuario dado, en este caso un menor de edad, se permite el acceso al sistema dependiendo de la hora en la que nos encontremos. Para ello, establece una política a aplicar obtenida de nuestro fichero policy.xml, y genera tanto una petición como una respuesta, evaluando el resultado.

Figura 31: Módulo *security manager*

Como hemos visto, muchas de estas clases interactúan con la capa de persistencia, con operaciones de lectura de datos. Pero existen otras clases como son la de `Logs` y `CrearPerfil` que se dedican exclusamente a la creación de nuevas entradas en los archivos de almacenamiento (véase en la figura 32).

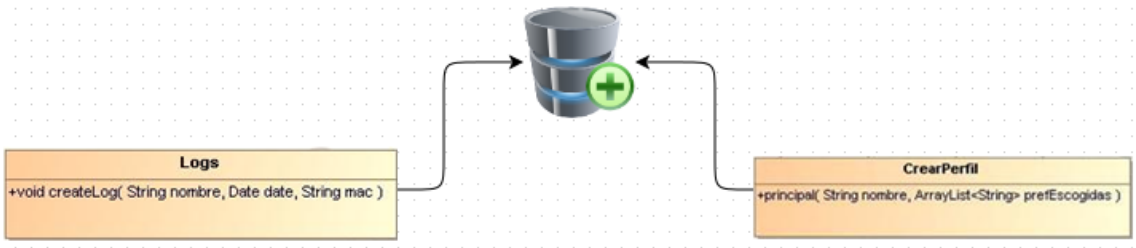

Figura 32: Clase *Logs* y *CrearPerfil*

7.3.2 Interfaces

Nada más lanzar el sistema, toda la lógica de negocio se pone en funcionamiento, y en la pantalla podemos observar una interfaz que nos da la opción de comenzar con el uso. En la figura 33 se observa el aspecto que presenta la pantalla de inicio, donde tras pulsar el botón OK comienza la detección de dispositivos, las comprobaciones acerca de las preferencias y las políticas de acceso, como podemos observar en la figura 34.

Figura 33: Pantalla de comienzo

Figura 34: Pantalla de búsqueda de dispositivos

Si no se encuentra ningún dispositivo activado, el sistema muestra una pantalla como la de la figura 35, y se mantiene detectando dispositivos con el fin de encontrar alguno y proseguir con la lógica. Esta pantalla es la misma que se muestra cuando tras la activación de un usuario, éste abandona el sistema, terminando con su sesión.

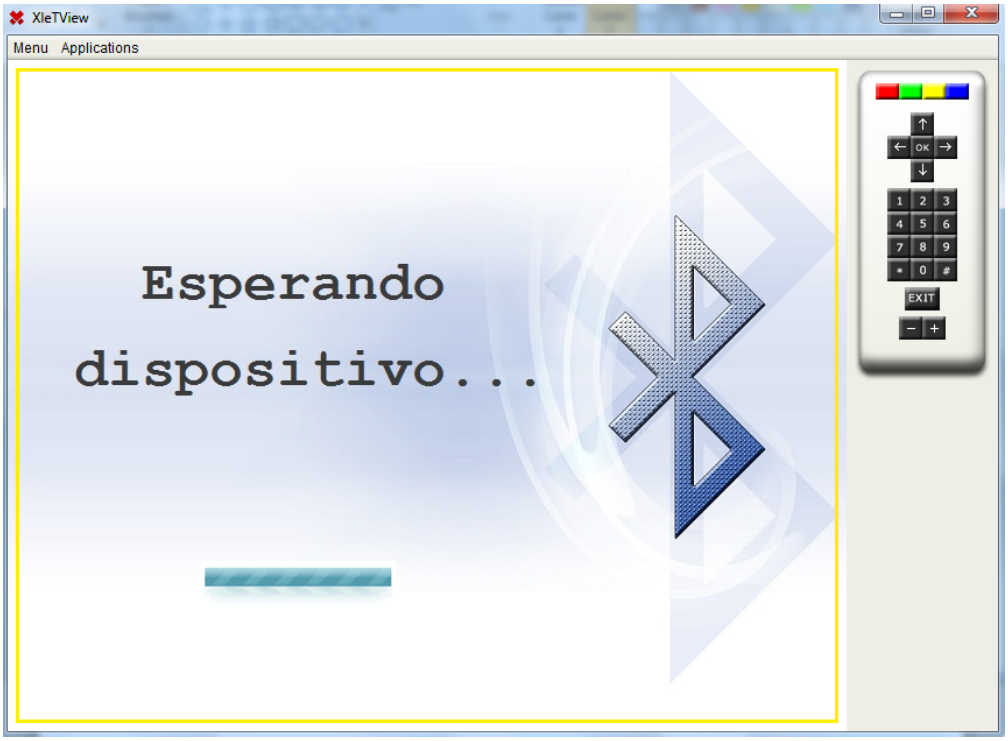

Figura 35: Pantalla de espera de dispositivos

Cuando un usuario se encuentra delante del televisor y se han llevado a cabo todas las comprobaciones pertinentes el sistema procede a la muestra por pantalla de una guía electrónica de programas adaptada a sus gustos. En la figura 36 podemos observar el aspecto que presenta una EPG personalizada de un usuario que se sienta delante del televisor.

Figura 36: EPG personalizada

Si en el momento que se active el sistema, se detecta algún dispositivo o dispositivos que no estén registrados, se iniciará la interfaz de registro de la aplicación web. Una vez se registre el primero de ellos, se cargaran sus gustos de acuerdo a sus preferencias. Si en ese momento registramos el resto de dispositivos, el sistema aplicará las preferencias de grupo, basadas en la comparación de éstas para establecer gustos comunes.

Es posible que entre los usuarios se encuentre la presencia de un menor, en un horario inadecuado. En este caso, la política de acceso realizará su función, bloqueando el sistema y mostrando una pantalla como la de la figura 37.

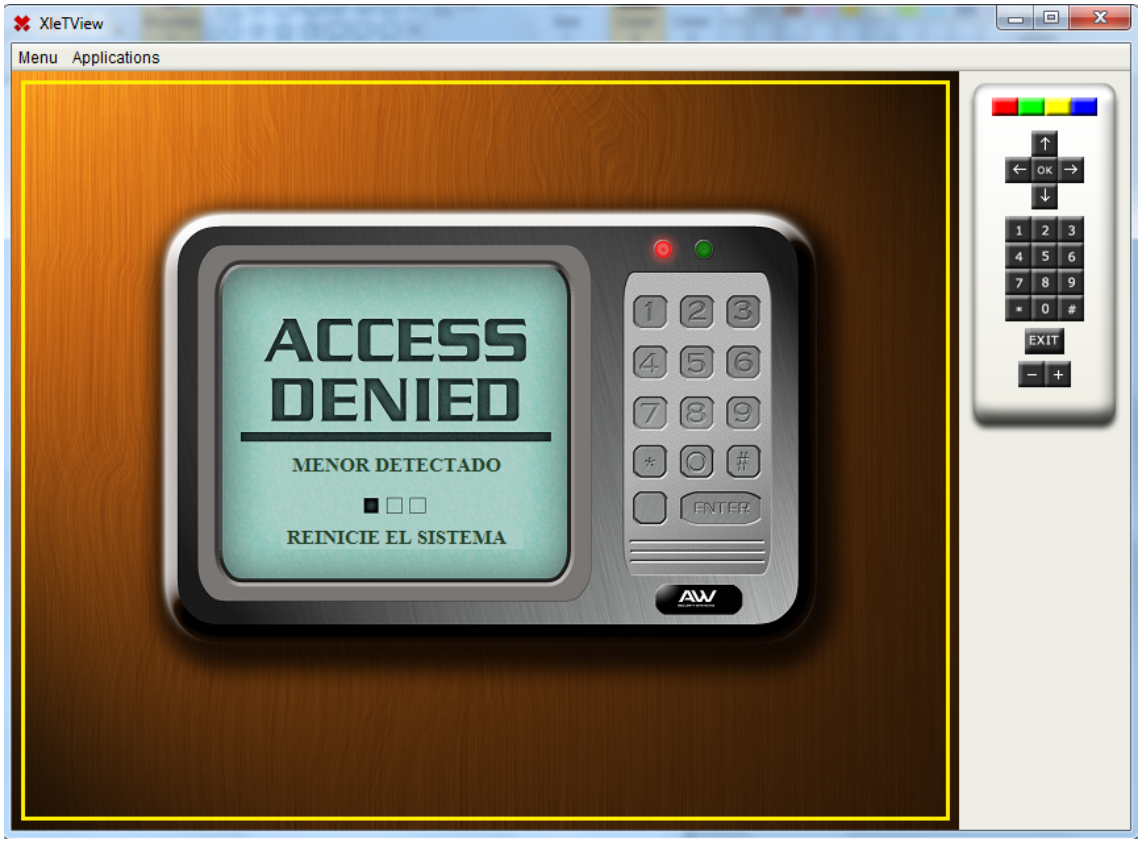

Figura 377: Pantalla de acceso denegado

7.4 Almacenamiento de datos

Para el almacenamiento de entradas en el sistema, como de preferencias de usuarios o políticas de acceso, el sistema hace uso de archivos tanto XML como de texto. Ya vimos en la parte de los *servlets* como estaban estructurados los diferentes archivos necesarios para la lectura o creación de prioridades y cómo lo hacían los distintos programas en el archivo que emulaba la EPG que podemos encontrar en cualquier *set-top-box* actual.

En la figura 38, podemos observar un ejemplo de algunas entradas realizadas en el sistema durante la sesión de pruebas. En ellas, se puede observar la hora de entrada en el sistema, la MAC del dispositivo y el nombre del usuario, que corresponde con el identificador.

```
Usuario: Javier - Mac: BC851F553EAA - Fecha: Mon Jun 04 19:02:20 CEST 2012
Usuario: Maria - Mac: FC854F553F26 - Fecha: Mon Jun 04 19:03:00 CEST 2012
Usuario: Javier - Mac: BC851F553EAA - Fecha: Wed Jun 06 09:40:33 CEST 2012
```

Figura 38: Fichero de logs

La parte de la política de acceso, la cual está gestionada por el modulo SM, esta implementada mediante tres archivos:

- Policy.xml: contiene la lógica de acceso al sistema con los parámetros definidos para tal función, como se refleja en la imagen 39.

```
<Rule RuleId="LoginRule" Effect="Permit">

  <!-- Only use this Rule if the action is login -->
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">login</AttributeValue>
          <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
                                     AttributeId="ServerAction"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>

  <!-- Only allow logins from 07:00 to 23:00 EDT -->
  <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
        <EnvironmentAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#time"
                                        AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
      </Apply>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">07:00:00</AttributeValue>
    </Apply>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
        <EnvironmentAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#time"
                                        AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
      </Apply>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">23:00:00</AttributeValue>
    </Apply>
  </Condition>

</Rule>

<!-- We could include other Rules for different actions here -->

<!-- A final, "fall-through" Rule that always Denies -->
<Rule RuleId="FinalRule" Effect="Deny"/>
```

Figura 39: Fichero policy.xml

- Request.xml: petición de acceso en formato XACML, como se refleja en la imagen 40.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request xmlns="urn:oasis:names:tc:xacml:1.0:context" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context cs-xacml-schema-context-01.xsd">
  <Subject />
<Resource>
<Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="http://www.w3.org/2001/XMLSchema#string">
  <AttributeValue>SampleServer</AttributeValue>
  </Attribute>
  </Resource>
<Action>
<Attribute AttributeId="ServerAction" DataType="http://www.w3.org/2001/XMLSchema#string">
  <AttributeValue>login</AttributeValue>
  </Attribute>
  </Action>
  </Request>
```

Figura 40: Fichero request.xml

- Respose.xml: respuesta a la petición en formato XACML, como se refleja en la imagen 41.

```
<Response>
<Result>
  <Decision>Permit</Decision>
<Status>
  <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok" />
  </Status>
  </Result>
  </Response>
```

Figura 41: Fichero response.xml

Capítulo 8. Pruebas de rendimiento

En este capítulo, se pretende determinar si el sistema realizado es viable o no. Para ello, vamos a realizar una serie de escenarios de pruebas, que nos determinen si cumple con la funcionalidad para la cual ha sido diseñado y llegar así a una conclusión.

El sistema está diseñado para adaptar el contenido de la televisión a las preferencias y gustos de los distintos usuarios registrados y activos. Es por lo tanto que debe responder de diferentes maneras dependiendo del número de usuarios presentes y de su edad, para activar políticas de acceso o establecer preferencias comunes. Además, la aplicación Web tendrá que comportarse de acuerdo al módulo que representa, determinando el registro, la modificación y la eliminación de los perfiles según el caso. A continuación vamos a listar una serie de escenarios de pruebas, como ilustra la tabla 2, para ver cómo responde la aplicación a cada uno de ellos y posteriormente obtendremos una conclusión que nos permitirán evaluarlo.

8.1 Escenarios de pruebas

8.1.1 Aplicación Web

Prueba realizada	Resultado	Observaciones
Registro de un perfil	OK	Los datos son recogidos y guardados en un fichero y se ejecutan las acciones correspondientes. Si el fichero no existe, se crea. Si el perfil ya está registrado, lo notifica con un mensaje por pantalla.
Eliminación de un perfil	OK	Se muestran los perfiles registrados para la eliminación de uno de ellos.
Modificación de un perfil	OK	Los datos son recogidos y se ejecutan las acciones correspondientes.
Campos editables	OK	Se muestran los valores almacenados y se permite su modificación.
Detección de usuarios	OK	Se realiza una búsqueda de dispositivos y se listan.
Cambios de pantallas	OK	La navegación por las pantallas de la aplicación responde al modelo diseñado.

Tabla 2: Pruebas de la aplicación Web

Con las pruebas de la tabla 2, verificamos que las tres se ajustan a la funcionalidad para la cual han sido creadas. Además, comprobamos también que la estructura del fichero que almacena perfiles una vez realizada alguna de estas acciones se mantiene, objetivo fundamental para que el filtrado del archivo XML se produzca correctamente.

8.1.2 XletView

Prueba realizada	Resultado	Observaciones
El sistema comienza su funcionamiento con dispositivos activos en su rango de acción, pero ninguno de ellos se corresponde con los registrados en los ficheros de almacenamiento de datos.	OK	El sistema detecta que ninguno de los dispositivos activos ha sido registrado con anterioridad en el sistema, por lo que automáticamente se ejecutaría la aplicación Web para permitir el registro de estos nuevos dispositivos para que comiencen así a disfrutar del sistema. Una vez registrado todos, se lleva a cabo una adaptación automática del contenido de la televisión a las preferencias comunes. Si no pretendemos registrarlos todos, deberemos apagar el Bluetooth de los dispositivos que no queramos.
El sistema comienzo su funcionamiento con dispositivos activos en su rango de acción, entre los que se encuentra un niño, pero ninguno de ellos se corresponde con los registrados en los ficheros de almacenamiento de datos.	OK	Al ejecutar la aplicación web y terminar con el registro de los dispositivos que queramos, el sistema detecta la presencia de un menor entre los nuevos registros, por lo que automáticamente ejecuta la política de acceso, bloqueando el sistema si nos encontramos en la franja horario de las 23:00 a las 07:00. Para relanzar el sistema, deberemos de desactivar el dispositivo del niño y ejecutar la aplicación de nuevo, si no queremos que la política de acceso vuelva a saltar.
El sistema comienza su funcionamiento con un dispositivo activo y registrado en los ficheros de almacenamiento de datos.	OK	El sistema detectaría tan solo un usuario, por lo que no sería necesario hacer una búsqueda de gustos comunes. Simplemente adapta el contenido de la televisión a las preferencias del usuario para que comience a disfrutar de la aplicación.
El sistema comienza su funcionamiento con varios dispositivos activos y registrados en los ficheros de almacenamiento de datos.	OK	Caso más común junto con la posibilidad de que sea tan solo un usuario el que se encuentra disfrutando del sistema. Se hará un filtrado por preferencias y gustos comunes a todos ellos, para posteriormente adaptar el contenido de la televisión a ellos. Al igual que en el aparatado 2, si alguno de ellos fuese un niño, automáticamente se cargaran sus preferencias. De esta manera se fomentaría que los padres viesen la televisión con sus hijos y además, que los más pequeños pudieran ver una programación inadecuada.

Tabla 3: Pruebas integradas XletView 1

Prueba realizada	Resultado	Observaciones
El sistema comienza su funcionamiento con varios dispositivos activos y registrados, siendo más de uno niños.	OK	El sistema no realiza una combinación de las preferencias y gustos de ambos niños, sino que simplemente adapta el contenido de la televisión al primero que detecte activo y registrado.
El sistema está funcionando con un usuario activo y registrado en los ficheros de almacenamiento, y acto seguido, se desactiva.	OK	Al hallarse solamente un usuario en el sistema y abandonarlo, el sistema se pondrá en modo búsqueda de dispositivos, para una vez detectados mas usuarios, seguir con la lógica de la aplicación y adaptar el contenido de la televisión una vez más.
El sistema está funcionando con varios usuarios activos y registrados en los ficheros de almacenamiento, y acto seguido, alguno de ellos se desactiva.	OK	Como aún quedarían uno o varios usuarios activos en el sistema, se mantendrán las preferencias y gustos comunes que se establecieron en un primer momento para todos. Si se desease obviar las preferencias del usuario que abandono, se deberá reiniciar el funcionamiento del sistema para que comience de nuevo con el filtrado de gustos.
El sistema está funcionando con uno o varios usuarios activos y registrados en los ficheros de almacenamiento, y acto seguido, otro usuario se desea incorporar.	OK	Se debería de volver a lanzar la aplicación de nuevo, ya que la adaptación del contenido de la televisión ha sido realizada para los usuarios que había activos y registrados, no para los nuevos que deseasen incorporarse. Una simple ejecución de nuevo de la aplicación descubriría a estos nuevos usuarios y realizaría una combinación de todas las preferencias y gustos, readaptando el contenido de la televisión.

Tabla 4: Pruebas integradas XletView 2

8.2 Conclusiones

Se ha realizado una batería de pruebas que se adaptasen a la totalidad de casos que se podían dar en el sistema, desde la detección simple de dispositivos sin registrar y registrados hasta a la modificación del numero de ellos mientras el sistema está funcionando.

Tras la realización de la batería de pruebas descrita en el apartado anterior, y tras comprobar que los resultados han sido los esperados de acuerdo a como ha el sistema ha sido implementado, se puede determinar que éste cumple con los objetivos especificados.

Bloque IV

Capítulo 9. Historia del proyecto

Para realizar este proyecto, se han empleado aproximadamente 8 meses. En este apartado se realiza un repaso de cómo se ha distribuido el trabajo en este periodo de tiempo y las fases seguidas, así como el tiempo aproximado empleado en cada una de ellas.

1. Estudio y análisis de la arquitectura del proyecto FamTV, 2 meses

Se ha llevado a cabo un estudio en profundidad de la arquitectura y el diseño de un proyecto propuesto denominado "*FamTV: Presence-Aware Personalized Television*". La implementación partió de cero, puesto que se trataba de tan solo un esquema de la arquitectura de este sistema, por lo que algunas decisiones fueron tomadas a medida que el proyecto iba avanzando. Algunas de las complicaciones surgieron a raíz de cómo obtener por ejemplo una EPG de un flujo MPEG-2, lo cual solucionamos tomando directamente un archivo que nos proporcionaba una guía electrónica de programas de una televisión digital de un país durante 7 días. Esto facilito mucho las cosas, ya que nos encontrábamos en un punto en el que nos era muy difícil avanzar.

2. Elección de las tecnologías para dar soporte al proyecto, 1 mes

Una vez determinadas las necesidades de nuestro proyecto, se barajaron las posibles tecnologías que mejor se adaptasen, como es el caso de Bluetooth, una aplicación web, XACML o el emulador de *set-top-box* XletView. En esta fase tuvimos ciertas dudas con el emulador a elegir, ya que debía estar orientado a implementaciones MHP por estar trabajando con flujos DVB. Disponíamos por lo tanto de *XletView*, *OpenMHP* o *MHP4free*. Pero el hecho de que el segundo no fuera tan completo y fácil de usar como el primero, y que la documentación *MHP4free* estuviera toda en alemán, nos hizo más simple la elección.

3. Diseño del sistema, 1 mes

El proyecto tiene que ser capaz de ofrecer un sistema de personalización de televisión con un detector de presencia. Para ello, se diseñaron los distintos módulos de acuerdo a la arquitectura analizada, para que cada uno realizara la funcionalidad requerida, como es el modulo de detección de presencia, el de seguridad, o el adaptador de contenido personal, entre otros

4. Implementación del sistema, 1 mes y medio

En esta fase del proyecto, se desarrolla el código de los distintos módulos según su funcionalidad. La aplicación Web ha necesitado de JSPs, *servlets* y *JavaScript* que corren sobre un servidor *Tomcat*, mientras que el resto de código han sido clases implementadas sobre la aplicación general, el Xlet que se ejecuta en el emulador, aportando la mayoría de la lógica de negocio del sistema.

5. Pruebas de rendimiento y puesta en marcha, 15 días

Para llevar a cabo esta fase, se han llevado a cabo distintas pruebas en las que se modificaba el número de usuarios en la sala, se introducía la figura del menor en horarios no permitidos para ellos, o se detectaba la presencia de un dispositivo como pudiera ser una impresora en alguna habitación cercana. A partir de esto, se ha determinado el grado de viabilidad del sistema así como los distintos resultados obtenidos.

6. Elaboración de la memoria, 1 mes y medio

En esta fase del proyecto, se ha redactado la memoria acorde con toda la documentación recopilada meses antes, la cual nos permitió la realización de todas y cada una de las fases.

Capítulo 10. Conclusiones y trabajos futuros

10.1 Conclusiones

El objetivo principal del proyecto se ha alcanzado, la creación de un sistema de personalización de guías de la actual televisión terrestre con detección de presencia, ofreciendo al usuario o usuarios la posibilidad de obtener de manera rápida y sencilla un resumen de los canales y programas que se están emitiendo en el momento de acuerdo a sus gustos y preferencias, sin tener que recorrer la guía electrónica de programas que se proporciona por defecto en cualquier *set-top-box* interno o externo. Y todo esto tan solo con sentarse delante del televisor de su casa.

Estudiando los distintos tipos de tecnologías inalámbricas de las que disponemos, llegamos a la conclusión de que es Bluetooth la que nos va a proporcionar una serie de características con las que va a ser posible satisfacer toda la funcionalidad planteada. Además del estudio de tecnologías inalámbricas, se lleva a cabo otro acerca de un emulador, XletView, lo suficientemente completo para poder ejecutar nuestra aplicación Java y disponer de los recursos necesarios para poder visualizar los resultados por pantalla, como si de nuestro televisor del salón se tratase.

Además, a través de una aplicación web que es capaz de correr sobre un servidor local, con las consiguientes ventajas a nivel de privacidad y ataques que esto supone, hemos dispuesto un servicio de registro, modificación y eliminación de perfiles de usuarios. Esto dota al sistema de una gran facilidad a la hora de comenzar con el uso del sistema, así como de su mantenimiento.

Una vez diseñado y desarrollado el sistema, se han llevado a cabo una serie de pruebas de rendimiento, que nos han proporcionado unos resultados satisfactorios en cuanto a número de usuarios y detección de nuevos dispositivos mientras el sistema está en uso se refiere.

10.2 Futuras líneas de trabajo

Como ya hemos comentado, nuestro sistema hace uso de la tecnología Bluetooth para la detección de dispositivos (usuarios). Esto supone que se deben encontrar en el radio de acción de nuestro *set-top-box*, el cual, aparte de los usuarios de nuestro sistema, puede detectar otros dispositivos como impresoras, o videoconsolas que se encuentren en habitaciones cercanas.

Es por ello que una futura línea de trabajo sería mejorar este sistema de detección, con alguna **otra tecnología de detección**, o con la incorporación de elementos que nos ayuden a cubrir la totalidad de inconvenientes que se pueden dar, como es el caso de una cámara que visualice e identifique a los usuarios.

Además, puesto que si se dispone de un *set-top-box* se puede actualizar la EPG a nuestro gusto siempre que queramos**, se podría trabajar directamente con ella**, sabiendo que siempre la íbamos a encontrar actualizada.

En cuanto a la política de filtrado para varios usuarios, se podría llevar a cabo una mejora en cuanto al sistema que se usa, **aunando los vectores prioridad de los géneros** de los distintos usuarios para sintetizarlos en uno, y a tras una lista de canales comunes a ambos, mostrar la programación acorde.

Para finalizar, se podría hacer uso de **la librería Log4j** para mostrar mensajes de información de las acciones que el sistema va realizando, como errores, mensajes o avisos en tiempo de ejecución. De esta manera dispondríamos de un control de los eventos pudiendo guardarlos en un archivo de texto, mostrándolos por pantalla, etc.

Bloque V

Apéndices

Apéndice A. Presupuesto

1. Autor: Alvaro Moscoso Cabrera

2. Departamento: Ingeniero Telemática

3. Descripción del proyecto:

   - Título: **Diseño e implementación de un sistema de personalización de TV con detector de presencia.**
   - Duración (meses): **7**
   - Tasa de costes indirectos: **20%**

4. Presupuesto total del proyecto (valores en Euros): 12.400 Euros

5. Desglose presupuestario (costes indirectos):

Apellidos y nombre	Categoría	Dedicación (hombres mes*)	Coste hombre mes	Coste (€)
Almenares Mendoza, Florina	Ingeniero SR	0,2	4.289,54	857,91
Moscoso Cabrera, Alvaro	Ingeniero JR	3,5	2.694,39	9.430,36
			Total	**10.288,27**

* 1 Hombre mes = 131,25 horas. Máximo anual de dedicación 12 hombres mes (1.575 horas). Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

Descripción	Coste (€)	% Uso dedicado al proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable **
PC con WinXP, office y Bluetooth	850	100	7	60	99,16
Móvil con Bluetooth	100	100	5	60	8,33
				Total	**107,49**

- Fórmula de cálculo de la amortización:

(**A**/**B**) x **C** x **D**

A = número de meses desde la fecha de facturación en que el equipo es utilizado

B = periodo de depreciación (60 meses)

C = coste del equipo (sin IVA)

D = % del uso que se dedica al proyecto (habitualmente 100%)

6. Resumen de costes:

	Presupuesto costes totales
Personal	10.288,27
Amortización	107,49
Costes indirecto	2079
Total	**12474**

Apéndice B. Instalación y configuración de herramientas

B.1 XletView

La aplicación consiste en utilizar el emulador del *set-top-box* para Ginga-J, para ejecutar una aplicación, la cual mostrara en la pantalla un texto inicial, y enseguida este texto, cambiara para otro, equivalente al botón presionado por el control remoto.

Para la ejecución de un aplicativo Xlet, es necesario tener instalado los siguientes programas y librerías:

- Java 2 *Standard Development Kit*: proporciona el ambiente de ejecución para el desarrollo Java [21].

- IDE (*Integrated Development Enviroment*) es un ambiente integrado de desarrollo, y un programa de computador que reúne las características y herramientas de apoyo al desarrollo de software con el objetivo de agilizar este proceso. EN este caso haremos uso de Eclipse [22].

- Java TV API nos ofrece un conjunto de clases e interfaces a fin de proveer funcionalidades y servicios interactivos de TV para el set-top-box [23].

- XletView (xletview-o.3.6) contiene el emulador para las aplicaciones Xlet [24].

Debido a que el XletView solo acepta archivos "`.class`" (Xlets), para ser ejecutados, se utilizara la librería "`XletView.jar`", que incluye recursos de la plataforma HAVi (*Home Audio Video Interoperability*). El "`XletView.jar`" es una librería Java de componentes gráficos desarrollados especialmente para la creación de interfaces para ambientes de TV. Para testear las aplicaciones haremos uso de la clase `HscenseFactory` y `Hscene` del API HAVi. El IDE Eclipse convierte los archivos "`.java`" en archivos "`.class`", que son lo que van a utilizar. En la Figura 42 se ilustra, como las librerías "`javatv.jar`" y "xletview.jar" son incluidas en el "*Build Path*" de Java.

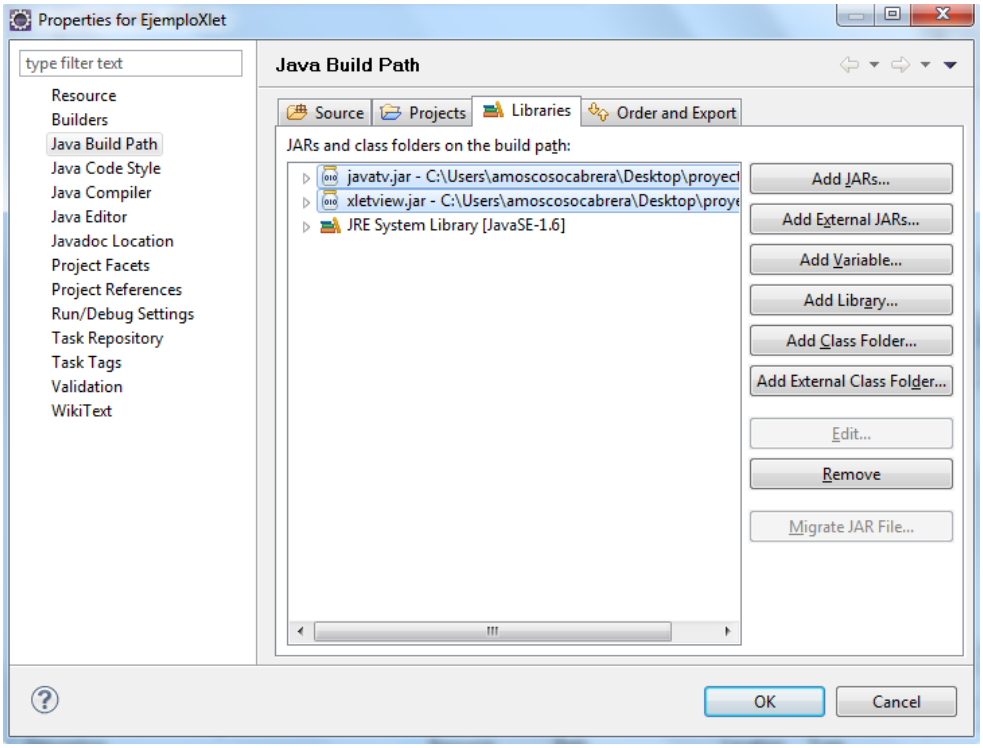

Figura 42: IDE Eclipse: *Java Build Path* [9]

Para realizar el teste con el emulador, se ejecuta el "XletView.jar", que se encuentra en el directorio donde fue descargado, haciendo doble clic sobre el archivo, o escribiendo por consola: `java –jar xletview.jar`.

Para cargar un Xlet en el emulador del *set-top-box*, se debe llamarlo dentro de la carpeta *Default Group*, un directorio nativo de XletView. Así también, se puede crear otro subdirectorio haciendo clic en *New Group*, construyendo un grupo particular de Xlets. Enseguida, se cargar la Xlet, dentro del *Default Group*, haciendo clic en *New Application*, luego en *new app 1*, con lo que aparecerá en el lado derecho tres cajas de texto, las cuales deben ser editadas para la ejecución del Xlet.

- *Name*: Nombre del proyecto, con que la Xlet se mostrara en el menú aplicaciones.

- *Path* (ruta): Directorio donde se encuentra la Xlet.

- Xlet: Archivo "`.class`" de la aplicación que será cargada.

- A continuación, se procede a cambiar el nombre del proyecto si es necesario, indicar la ruta donde se encuentra el archivo "`.class`" y por último el nombre del archivo, como observamos en la figura 43. Finalmente, hacer clic en el botón *save & close*.

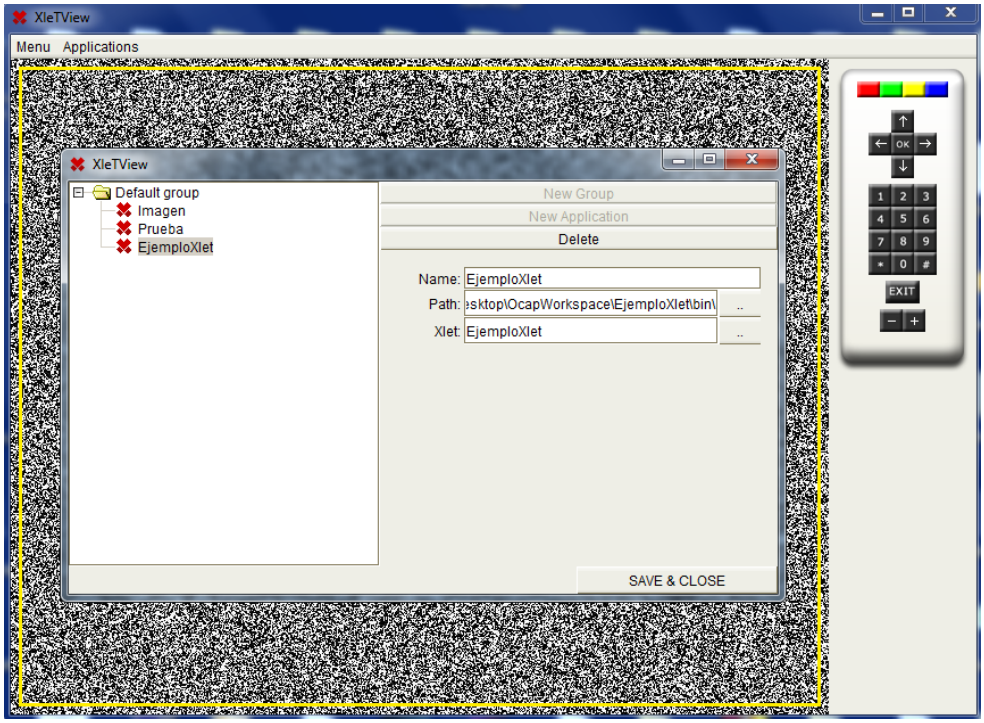

Figura 43: Pantalla con el nombre y ruta del Xlet [9]

Después de realizar la configuración, se procede a ejecutar la aplicación, haciendo clic en el menú *Applications*, el cual despliega una lista de las aplicaciones cargadas. Se hace clic en la aplicación escogida, en este caso la aplicación con el nombre "`EjemploXlet`".

En la figura 44 se ilustra la pantalla inicial de la aplicación, en la cual muestran dos textos "Hola Mundo Java" y "Control Remoto". En esta aplicación, el texto "Control Remoto", cambiara a otro texto conforme se presione un botón del control remoto. Por ejemplo, si se presiona el botón número 2 del control remoto, el nuevo texto, será: "Botón Numérico: 2", como se ilustra a continuación. Para regresar a la pantalla inicial de la aplicación, hacer clic en *Applications*, y escoger la opción *Reload Current*, reiniciando la aplicación a su estado inicial. Para cerrar el emulador haga clic en Menú y luego escoger *Exit* Alt+F4.

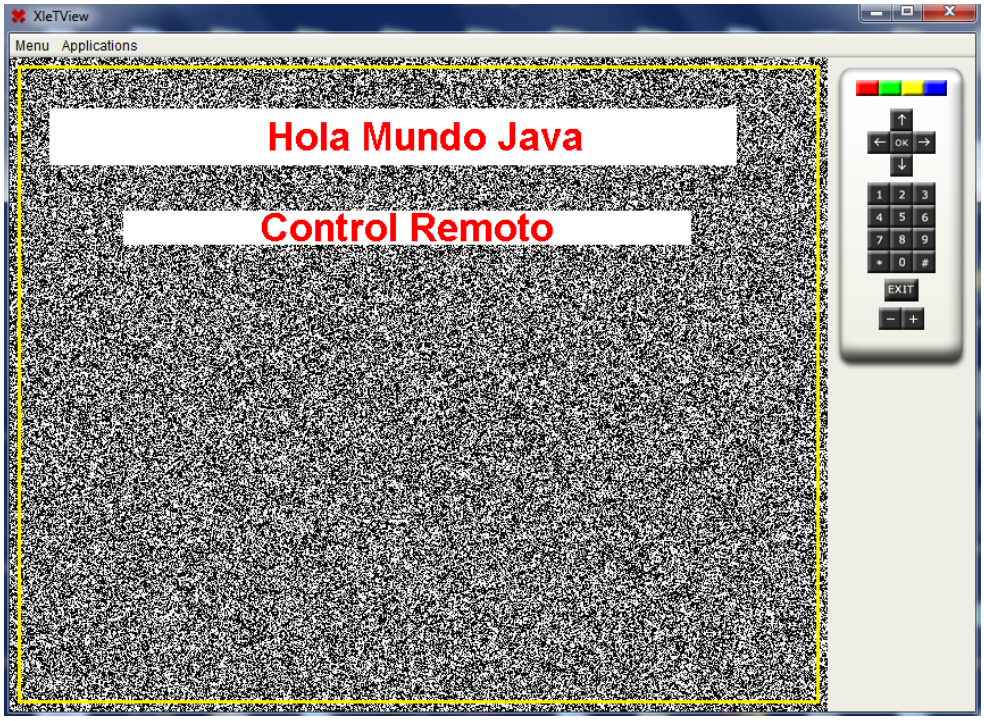

Figura 44: Visualización inicial de la aplicación al ser ejecutada [9]

En la figura 45 se puede observar lo que sucede cuando se pulsa un botón.

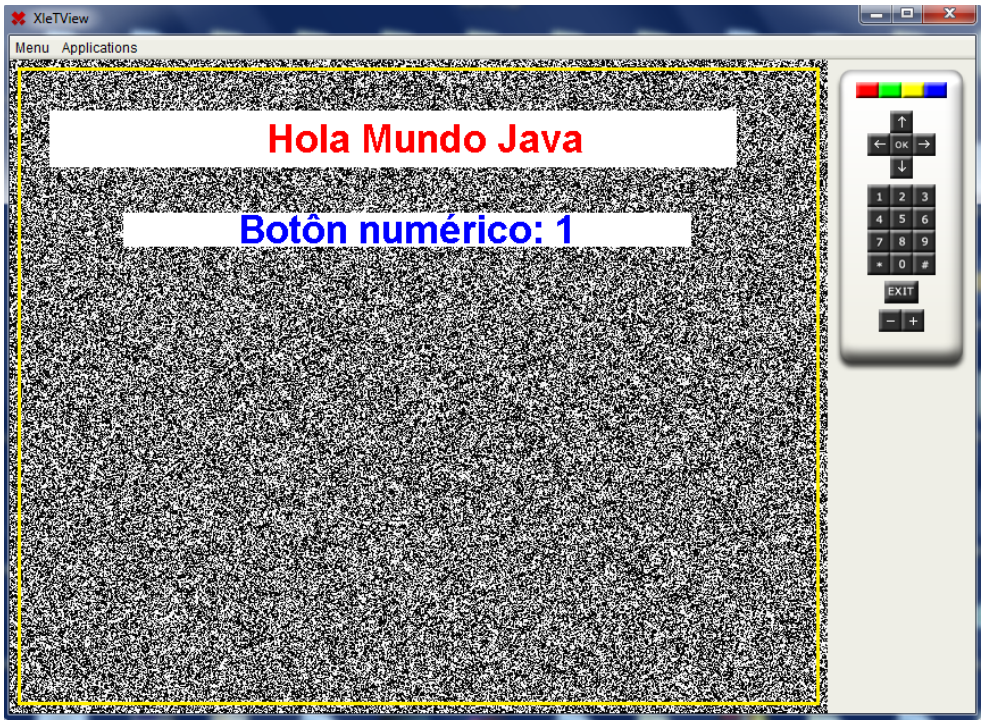

Figura 45: Pantalla con el nombre del botón pulsado [9]

B.2 Código `EjemploXlet`

```
import java.awt.*;
import java.awt.event.*;
import javax.tv.xlet.*;
import org.havi.ui.*;
import org.havi.ui.event.*;

public class ExemploXlet implements Xlet, KeyListener {

      private XletContext context;
      private HScene scene;
      private HStaticText label1, label2;

      public ExemploXlet() {

      public void initXlet(XletContext xletContext)throws
      XletStateChangeException {

            /* guardando el contexto... */
            this.context = xletContext;

      }

      public void startXlet()throws XletStateChangeException{


            HSceneFactory hsceneFactory =
            HSceneFactory.getInstance();
            scene=hsceneFactory.getFullScreenScene(HScreen.getDefaul
            tHScreen().getDefaultHGraphicsDevice());scene.setSize(64
            0, 480);

            scene.setLayout(null);scene.addKeyListener(this);
            //el própio Xlet es el oyentelabel1 = new
            HStaticText("Hola Mundo Java", 35, 45, 660, 50,new
            Font("Tiresias", 1, 36),Color.red, Color.white,new
            HDefaultTextLayoutManager());

            label2 = new HStaticText("Control Remoto", 100, 135,
            500, 30,new Font("Tiresias", 1, 36),Color.red,
            Color.white,new HDefaultTextLayoutManager());
```

```
      scene.add(label1);scene.add(label2);
      scene.setVisible(true);
      scene.requestFocus();

}
public void pauseXlet() {

/* Método vacio */

}

public void destroyXlet(boolean unconditional)throws
XletStateChangeException {

      if (scene!=null) {

            scene.setVisible(false);
            scene.removeAll();
            scene = null;

      }
      context.notifyDestroyed();
}

/* Método de java.awt.event.KeyListener */
public void keyTyped(KeyEvent keyevent) {

      /* Método vacio */

}

/*Método de java.awt.event.KeyListener */
public void keyReleased(KeyEvent keyevent) {

      /* Método vacio */

}

/* Método de java.awt.event.KeyListener */
public void keyPressed(KeyEvent e) {

      String mensagem = "";
      int codigo = e.getKeyCode();
```

```
switch (codigo) {

case 48:
case 49:
case 50:
case 51:
case 52:
case 53:
case 54:
case 55:
case 56:
case 57:  mensagem += "Bot\u00F4n num\u00E9rico:
"+(codigo-48);
          break;
case 403:  mensagem += "Bot\u00F4n Rojo";
          break;
case 404:  mensagem += "Bot\u00F4n Verde";
          break;
case 405:  mensagem += "Bot\u00F4n Amarillo";
          break;
case 406:  mensagem += "Bot\u00F4n Azul";
          break;
case 27:  mensagem += "Bot\u00F4n EXIT";
          break;
case 10:  mensagem += "Bot\u00F4n OK";
          break;
case 151:  mensagem += "Bot\u00F4n Asterisco (*)";
          break;
case 520:  mensagem += "Bot\uu00F4n Michi (#)";
          break;
case 38:  mensagem += "Arriba";
          break;
case 40:  mensagem += "Abajo";
          break;
case 37:  mensagem +=Izquierda";
          break;
case 39:  mensagem += "Derecha";
          break;
default:mensagem += "Hola Mundo Java";

}
```

```
            label2 = new HStaticText(mensagem, 100, 135, 500, 30,new
            Font("Tiresias", 1, 36),Color.blue, Color.white,new
            HDefaultTextLayoutManager());

            scene.removeAll();
            scene.add(label1);
            scene.add(label2);
            label2.repaint();
            scene.repaint();

      }

}
```

Glosario

ATSC (*Advanced Television System Committe*): grupo encargado del desarrollo de los estándares de la televisión digital en los Estados Unidos.

Bouquet: grupo de servicios ofrecidos por un mismo proveedor. En la terminología comercial utilizada en España este concepto se ha traducido por paquete (p.ej. paquete básico).

DTMB (*Digital Terrestrial Multimedia Broadcast*): es el estándar de Televisión para terminales fijos y móviles utilizado en la República Popular China, Hong Kong y Macao.

ISDB-T (*Integrated Services Digital Broadcasting*): es un conjunto de normas creado por Japón para las transmisiones de radio digital y televisión digital.

DVB (*Digital Video Broadcasting*): es una organización que promueve estándares aceptados internacionalmente de televisión digital, en especial para HDTV y televisión vía satélite, así como para comunicaciones de datos vía satélite.

MPEG-2 (*Moving Pictures Experts Group*): es la designación para un grupo de estándares de codificación de audio y vídeo acordado por MPEG y usado para codificar audio y vídeo para señales de transmisión.

QAM (*Quadrature Amplitude Modulation*): es una técnica de modulación digital avanzada que transporta datos, mediante la modulación de la señal portadora de información tanto en amplitud como en fase.

QPSK (*Quadrature Phase-Shift Keying*): es una forma de modulación angular que consiste en hacer variar la fase de la portadora entre un número de valores discretos.

COFDM (*Coded Orthogonal Frequency Division Multiplexing*): es una técnica compleja de modulación de banda ancha utilizada para transmitir información digital a través de un canal de comunicaciones:

DASE (*DTV Application Software Enviroment*): norma desarrollada por el ATSC como estándar de los EE.UU. para l acapa de middleware de set top boxes de TV digital.

ARIB (*Association of Radio Industries and Businesses):* organización voluntaria que se encargada de regir, crear y mantener los estándares de emisión de la señal audiovisual en el área japonesa.

OCAP (*Open Cable Applications Platform*): especificación común para la capa middleware para los sistemas de cable en los Estados Unidos.

JMF (*Java Media Framework*): es una extensión de Java que permite la programación de tareas multimedia en este lenguaje de programación.

LDAP (*Lightweight Directory Access Protocol*): protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

HAVi (*Home Audio Video Interoperability*): arquitectura de software distribuida que especifica un conjunto de API diseñados para la interconexión directa de los aparatos de consumo de video y audio digital de diferentes tipos y proveedores.

DAVIC (*Digital Audio Video Council*): estándar cuyas APIs permitían a los objetos Java el acceso a algunos servicios de información, control de prestaciones de contenido de audio y video y el manejo de la gestión de los recursos en el receptor.

Ginga-NCL (Ginga Nested Context Language): módulo de la implementación Ginga que promueve la infraestructura de presentación de aplicaciones basadas en documentos hipermedia en lenguaje NCL.

Bibliografía

1. Héctor Cenador García, David Rincón**, "**DVB: Mecanismos de acceso condicional", Febrero 2006. Disponible en http://upcommons.upc.edu/pfc/bitstream/2099.1/11006/1/DVB_Mecanismos_de_acceso_condicional_-_Hector_Cenador_Garcia.pdf (consulta en Enero 2012).
2. Wikipedia: Televisión digital terrestre, Disponible en http://es.wikipedia.org/wiki/Televisi%C3%B3n_digital_terrestre, Enero 2012.
3. Carlos Cano Inés, Miguel Ángel González, Ignacio Pedraza, "Estudio de una solución para la medición de audiencias en la Televisión Digital Terrestre (TDT)", 2006. Disponible en http://eprints.ucm.es/8893/1/Memoria.pdf (consulta en Enero 2012).
4. Vanessa Cuesta, "MPEG". Disponible en: http://www.slideshare.net/antenared/mpeg-5326072, Febrero 2012.
5. Secretaría de Estado de Telecomunicaciones y para la Sociedad de la Información, Subgrupo de Normalización Técnica de la presentación, Grupo Técnico del Foro de la Televisión de Alta Definición en España, "Alta definición y acceso condicional", Abril, 2008. Disponible en: http://www.televisiondigital.es/TecnologiasRelacionadas/AltaDefinicion/ForoTVAD/ConclusionesForo/1SG1T4TABLASSIDETVADYACCESOCONDICIONAL.pdf (consulta en Febrero 2012).
6. Wikipedia: Guía electrónica de programas, Febrero 2012 http://es.wikipedia.org/wiki/Gu%C3%ADa_electr%C3%B3nica_de_programas
7. http://www.hdtvtest.co.uk/news/sony-kdl40v4000-review-20080822127.htm, Febrero 2012.
8. Wikipedia versión inglesa: *Electronic program guide*, Enero 2012 http://en.wikipedia.org/wiki/Electronic_program_guide.
9. Arquitectura hardware y software de un *set-top-box*, Febrero 2012 http://aat.inictel-uni.edu.pe/files/SET_TOP_BOX%28Informe_de_Avance1%29.pdf
10. http://es.wikipedia.org/wiki/Multimedia_Home_Platform, Marzo 2012
11. Xlet, Febrero 2012 http://bibing.us.es/proyectos/abreproy/11915/fichero/AP%C3%89NDICES%252FAp%C3%A9ndice_A_Xlets.pdf
12. http://bibing.us.es/proyectos/abreproy/11915/fichero/AP%C3%89NDICES%252FAp%C3%A9ndice_A_Xlets.pdf, Marzo 2012
13. API JavaTV, Febrero 2012 http://docs.oracle.com/javame/config/cdc/opt-pkgs/api/jsr927/index.html
14. Wikipedia: Bluetooth, Febrero 2012 http://es.wikipedia.org/wiki/Bluetooth
15. Alberto Gimeno Brieba." JSR-82: Bluetooth desde Java", 2004.
16. API Bluetooth, Marzo 2012 http://bluecove.org/bluecove/apidocs/index.html

17. Tim Moses. “eXtensible Access Control Markup Language (XACML) Version 2.0”, OASIS Standard, 1 Feb 2005.
18. http://www.info-ab.uclm.es/descargas/thecnicalreports/DIAB-05-01-2/Seguridad_en_Servicios_Web.pdf, Abril 2012.
19. Patricia Arias Cabarcos, Rosa Sánchez Guerrero, Florina Almenárez Mendoza, Daniel Díaz Sánchez and Andrés Marín Lopez. "FamTV: An architecture for Presence-Aware Personalized Television." *IEEE Transactions on Consumer Electronics*, vol.57, no.1, pp.6-13, February 2011
20. Wikipedia: JavaScript, Marzo 2012. http://es.wikipedia.org/wiki/Javascript
21. Java 2 Standard Development Kit, Enero 2012. http://java.sun.com/j2se/1.4.2/install.html
22. Eclipse IDE, Disponible en http://www.eclipse.org/, Enero 2012.
23. Java TV, Disponible en http://docs.oracle.com/javame/config/cdc/opt-pkgs/api/jsr927/index.html, Enero 2012.
24. XletView 0.3.6, Disponible en http://xletview.sourceforge.net/index.php, Enero 2012.